Studiengang Informationsverarbeitung Köln, den 15.02.2005
WS 2004/2005
Historisch-kulturelle Informationsverarbeitung
Magisterarbeit bei Prof. Dr. Manfred Thaller:

Die Verwendbarkeit von Game-Engines zur Analyse verlorener Räume

vorgelegt von

Peter Melms

Matrikelnummer: 294 58 94

E-Mail: p.melms@gmx.de

Tel.: 0221 356 75 72

Ägidius Strasse 41

50937 Köln

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Der Einsatz von Computern erleichtert auch in den Geisteswissenschaften in zunehmendem Maße die Präsentation und Analyse von Forschungsergebnissen. Im Bereich der Archäologie ist es beispielsweise denkbar, mithilfe von Computern und geeigneter Software, das 3D-Modell einer zerstören antiken Tempelanlage oder einer antiken Stadt zu erstellen, welches es dem Betrachter/Benutzer ermöglicht, *interaktiv* diese Welten zu erforschen. Benutzer könnten *virtuell* die Strassen von Pompeji besuchen. Den Möglichkeiten der Gestaltung werden dabei praktisch nur durch die Phantasie des *Entwicklers* oder durch die verwendete Hardware Grenzen gesetzt (vgl. Fernie and Richards: 2003, S.6). Die stetig fortschreitende Entwicklung auf dem Gebiet der 3D-Grafik stellt immer realistischere Instrumente der Darstellung zur Verfügung.

Ein wichtiger Motor dieses Entwicklungsprozesses ist die Computerspiel-Industrie. Hier werden erhebliche Finanzmittel aufgewendet, um immer realistischere virtuelle Welten für Spieler und Benutzer zu erschaffen. Die Performanz der jeweiligen Game-Engine trägt entscheidend zum kommerziellen Erfolg bei: Wie effizient geht sie mit Hardware-Ressourcen um? Denn nicht jeder Spieler kann einen High-End PC sein Eigen nennen. Ist die Engine also auch mit älteren Rechnern lauffähig, damit nicht von vorneherein potentielle Käufer ausgegrenzt werden? Welcher PC-Besitzer ist schon bereit, seinen Rechner aufzurüsten, um ein bestimmtes Spiel spielen zu können.
Gleichzeitig verändert sich die Leistungsfähigkeit der verwendeten Hardware. Vergleicht man die Polygonleistung heutiger High-End Grafikkarten mit deren Vorgängergeneration, zeigt sich nahezu eine Verdoppelung der Füllrate von 3.8 Milliarden texels/s. (Nvidia GeforceFX 5950 Ultra[1]) auf 6.4 Milliarden texels/s. (Nvidia Geforce 6800 Ultra[2]) und das bei immer besserer Darstellungsqualität.

Die zunehmende Leistungsfähigkeit der heutigen Hardware ermöglicht also die Erschaffung immer realistischerer Szenarien, von nahezu photorealistischer Grafik bis hin zu komplexen Physiksimulationen – in Echtzeit.

Berücksichtigt man diesen - von der Spiele-Industrie forcierten - technischen Fortschritt, so liegt es nahe, dieses Potential auch für wissenschaftliche Zwecke nutzbar zu machen.

---

[1] Quelle: Nvidia Webseite, 31.01.05: http://www.nvidia.com/page/fx_5700.html.
[2] Ebd., 31.01.05, http://www.nvidia.de/page/geforce_6800.html.

1.1 Themenstellung

Die vorliegende Arbeit untersucht die Verwendbarkeit von Game-Engines zur Analyse verlorener Räume.[3] In wie weit können aktuelle Game-Engines die Simulation und Analyse verlorenen Räume ermöglichen?

Anhand einer ausgewählten historischen Stätte von großer kultureller Bedeutung wird gezeigt, wie die Realisierung eines solchen Vorhabens aussehen kann. In einem ersten Schritt soll die zur Verfügung stehende Software auf ihre Eignung für dieses Vorhaben hin überprüft werden. Anschließend wird ein 3D-Modell erstellt, was in einem weiteren Schritt in eine Game-Engine integriert wird. Inwieweit sich diese Vorgehensweise zur Rekonstruktion und Analyse verlorener Räume eignet, wird im Fazit ausgewertet werden.

1.1.1 Untersuchungsgegenstand: Die Höhle von Altamira

Der verlorene Raum, der exemplarisch näher betrachtet wird, ist die steinzeitliche *Höhle von Altamira*.

Diese Höhle ist eine der ersten und wichtigsten Fundstätten paläolithischer Höhlenkunst. Sie liegt in Nordspanien, genauer in *Santillana del mar*, ca. 5 km von der Costa Real entfernt, in der Nähe von *Santander*. Die Höhle von Altamira und ihre Malereien werden auf ca. 14.000-15.000 v. Chr. datiert. Entdeckt wurde sie im Jahre 1875 von *Marcelino Sanz de Sautuola*. Die Zeichnungen und Malereien von Altamira gehören zu den wichtigsten und eindrucksvollsten Kunstwerken des *Jung-Paläolithikums*.[4] Die gesamte Höhle erstreckt sich über eine Länge von ca. 270 m.[5]

Einige der bekanntesten Malereien befinden sich im *Saal der polychromen Decke* (in der folgenden Abbildung *gelb* eingezeichnet):

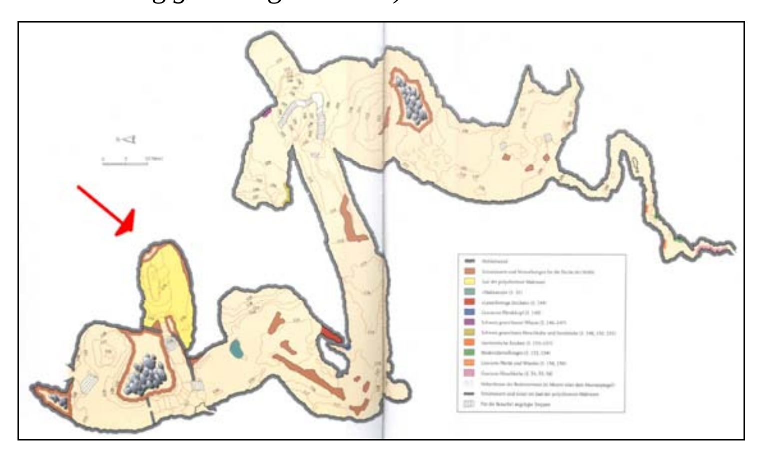


Abbildung 1: Saal der polychromen Decke.[6]

---

[3] Unter dem Terminus „verlorener Raum“ wird im Folgenden verstanden: Stätten von historisch-kultureller Bedeutung, die für die Öffentlichkeit nicht (mehr) zugänglich sind.

[4] Vgl. Antonio Beltrán: Altamira ... Mit Aufnahmen von Pedro A. Saura Ramos. Hrsg. und mit einem Vorw. von Gerhard Bosinski. Sigmaringen: Thorbecke,1998. S. 21.

[5] Vgl. Beltrán: Altamira, 1998. S. 20.

[6] Vgl. Beltrán: Altamira, 1998. S. 98-99.

1.1.2 Problemstellung

In Saal der polychromen Decke befindet sich eine Vielzahl an Zeichnungen, die stellenweise palimpsestartig angeordnet sind. Die scheinbare Bedenkenlosigkeit, mit der ältere Wandbilder durch jüngere überlagert werden, scheint frappierend und regt zu der Frage an, warum die Künstler diese nicht sauber voneinander getrennt angebracht haben.[7]

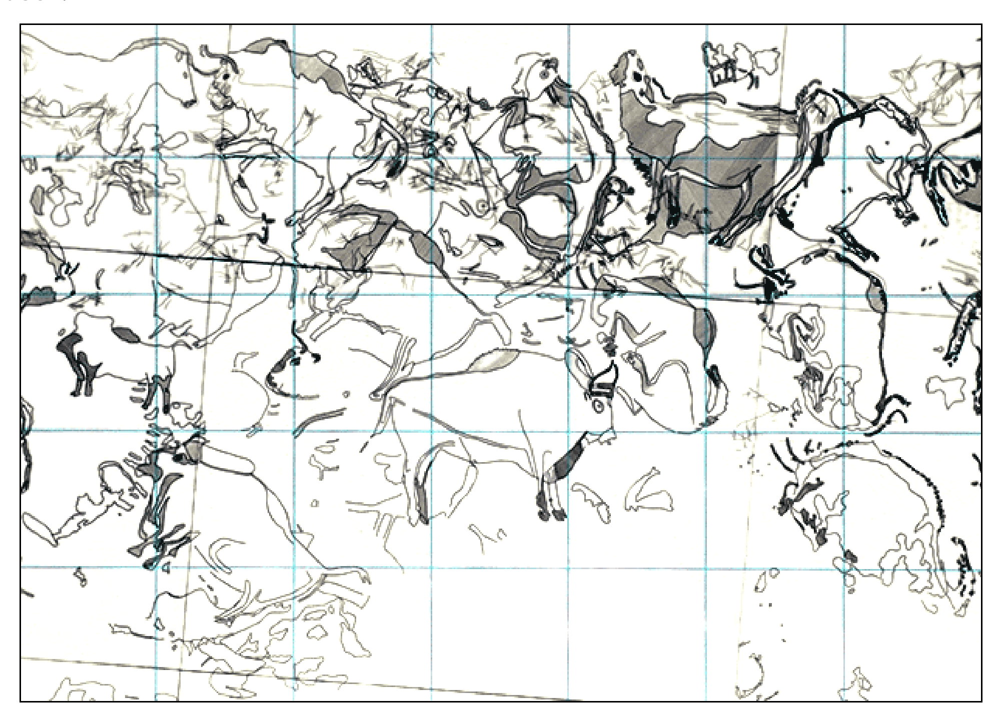


Abbildung 2: Abzeichnung der polychromen Decke[8]

Könnte die Beleuchtungssituation zur Zeit der Erstellung der Kunstwerke deren Anordnung beeinflusst haben? Wie hat der damalige Künstler die Höhlendecke wahrgenommen, wenn er z.B. mit einer Fackel oder beim Licht eines offenen Feuers gearbeitet hat?

Die Höhle ist für Besucher nicht mehr frei zugänglich. Die Klärung oben aufgeworfener Fragen kann durch einen Nachbau der Höhle, ihrer Deckenbeschaffenheit sowie der vorherrschenden Lichtverhältnisse erreicht werden. Hierzu wird ein 3D-Modell erstellt.

Mit der geeigneten Software können anschließend verschiedene Beleuchtungssituationen simuliert werden, die zeigen, wie stark die Sichtweite möglicherweise durch die damals zur Verfügung stehenden Lichtquellen eingeschränkt war.

Mit welcher Software lässt sich das bewerkstelligen? Ein 3D-Modelling-Programm allein eignet sich zwar dazu, ein hinreichend detailliertes Modell der Höhle zu erstel-

---

[7] Vgl. Biedermann, Höhlenkunst der Eiszeit, 1984. S. 48f.

[8] Quelle: 04.02.05, Webseite des Museums von Altamira: http://museodealtamira.mcu.es/ingles/museo/images/pinturas.gif.

len, hat aber den Nachteil, dass dieses Modell dann nur in vorberechneten Animationen betrachtet werden kann. Um die Auswirkung unterschiedlicher Lichtquellen auf die Umgebung sichtbar zu machen, müsste aber für jede Lichtquelle eine neue Animation erstellt werden, was einen relativ langwierigen Prozess darstellt. Zudem ist durch diese Art der Präsentation die *Interaktivität* mit dem Benutzer/Betrachter stark eingeschränkt.

Durch die bereits erwähnten Fortschritte im Bereich der Echtzeit-Computergrafik bietet sich für diesen Fall die Verwendung einer 3D-Game-Engine an, die in der Lage ist, das zuvor erstellte Modell der Höhle darzustellen und es darüber hinaus in Echtzeit interaktiv begehbar zu machen. Zudem muss die Engine in der Lage sein, Licht und Schatten dynamisch darzustellen. Ein Benutzer könnte sich so frei durch die Höhle bewegen und sie aus jedem Blickwinkel betrachten und untersuchen.
Nachdem nun ein Katalog mit Anforderungen formuliert ist, gilt es zu klären, welches 3D-Modellingtool und welche 3D-Engine für die Realisierung in Frage kommen.

1.1.3 Die verwendete Software: 3D-Modelling-Tools

Es stellt sich zunächst die Frage, welches Tool bei der Erstellung des Höhlenmodells Verwendung finden soll. Im professionellen Bereich sind folgende Programme am weitesten verbreitet: *3D Studio Max*[9] von *discreet*, *Cinema4D* von *Maxon*, oder *Maya* von *Alias-Wavefront*.
Maya ist vornehmlich auf die Animation von Körpern ausgerichtet. Es wurde für Animationsfilme wie „Ice Age“, aber auch für Special-Effects in „Herr der Ringe“ oder in „Star Wars: Episode 2“[10] eingesetzt. Zwar wird Maya auch in der Spiele-Entwicklung genutzt, aber auch hier vornehmlich zur Erstellung von animierten Figuren.
Cinema4D bietet, ähnlich wie *Maya*, eine Fülle an Tools für die Erstellung von Animationen, kann aber auch zur 3D-Modellierung verwendet werden. Es wird häufig für die Erstellung von Spezialeffekten in Film und Fernsehen eingesetzt.[11]
Eines der bekanntesten Programme ist *3ds Max*, vorliegend in Version 6. *3ds Max* ist ein sehr umfangreiches Programmpaket, für das viele zusätzliche Plugins und Hilfsprogramme existieren. Aufgrund seiner weiten Verbreitung hat sich eine breite Community von Nutzern gebildet, in deren Internetforen sich Lösungen für fast alle Probleme finden lassen.

---

[9] Kurz: 3ds Max.
[10] Quelle: 03.02.2005, Homepage von Alias-Wavefront: http://www.alias.com/eng/products-services/industries/feature_film/index.shtml.
[11] Quelle: 04.02.05, Homepage von Maxon: http://www.maxon.de/pages/dyn_files/dyn_htx/htx/sol_movie_d.html.

Gerade weil es im Bereich der Spieleentwicklung häufig eingesetzt wird, bieten die meisten Entwicklungsumgebungen für Spiele eine entsprechende Schnittstelle an, um *3ds Max* Dateien zu importieren, was dem Benutzer den Datentransfer zwischen den verschiedenen Applikationen entschieden erleichtert.

Bei der Entscheidung nicht zu vernachlässigen ist schließlich der finanzielle Aspekt, da die genannten Tools in der Vollversion nicht selten zwischen 3.000 und 4.000 Euro oder mehr kosten. Je nach Einsatzgebiet stehen aber auch günstigere Lehr- oder Studentenversionen zur Verfügung, die für nur einen Bruchteil des regulären Preises verkauft werden, ohne in ihrer Funktionalität eingeschränkt zu sein. Diese Versionen dürfen nicht kommerziell genutzt werden. In den meisten Fällen stehen zur Entscheidungshilfe zusätzlich zeitlich begrenzt nutzbare *Trial-Versionen* zur Verfügung, die über das Internet erhältlich sind.

Die Entscheidung für die Verwendung von *3ds Max* in dieser Arbeit resultiert in erster Linie aus oben genannten Gründen. Die gute Anbindung an eine Vielzahl von Game-Engines und Editoren und die breite User-Community sind aber nicht allein entscheidend. Letztlich ist die Auswahl aber immer auch eine Frage des persönlichen Geschmacks.

1.1.4 Die verwendete Software: Die Game-Engine

Die Wahl der richtigen Game-Engine wird erschwert durch die große Zahl der zur Verfügung stehenden, auch frei erhältlichen, 3D-Bibliotheken und Engines.
So sind z.B. viele Programmbibliotheken erhältlich, meist in C oder C++ geschrieben, die vom Benutzer erst in eine lauffähige 3D-Engine verwandelt werden müssen und so viel Programmierarbeit verlangen. Da dies für die vorliegende Arbeit zu zeitaufwendig gewesen wäre, wurden diese Bibliotheken bei der Suche von vorneherein nicht berücksichtigt.[12]

Im kommerziellen Spielebereich gibt es einige Spiele, die ihren eigenen Level-Editor mitbringen. Mit Hilfe dieses Editors ist ein Benutzer in der Lage, seine eigenen Level oder Maps zu gestalten. Außerdem ist es möglich, eigene Modelle in ein bestehendes Spiel zu integrieren.
Aber das ist nicht die einzige Fähigkeit, die der Editor und die Spiele-Engine zur Verfügung stellen müssen. Im zu untersuchenden Fall wird eine Grafik-Engine benötigt, die eine möglichst realistische Grafik bei guter Performance bietet. Dazu kommt, dass

---

[12] Beispiele hierfür sind: **Genesis3D** (http://www.genesis3d.com/), die **Irrlicht-Engine** (http://irrlicht.sourceforge.net/index.html) oder die **OGRE-Engine** (http://www.ogre3d.org/).

sie in der Lage sein muß, Licht und Schatten dynamisch, d.h. zur Laufzeit des Spiels zu berechnen und darzustellen, damit später das Höhlenmodell in verschiedenen Beleuchtungssituationen untersucht werden kann.
Neben der meist guten Anbindung des mitgelieferten Editors an bekannte 3D-Modellingtools, die dem Benutzer den Import eigener Modelle und so die Erstellung eigener Maps erleichtern, besteht ein weiterer Vorteil bei einer kommerziellen Spiele-Engine im regelmäßigen Support und der kontinuierlichen Weiterentwicklung der Engine durch die Entwickler des Spiels. Eine wichtige Rolle spielt außerdem der Bedienkomfort des mitgelieferten Editors. Wie leicht oder wie schwer gestaltet sich der Zugang für den Benutzer? Wie viel Einarbeitungszeit ist nötig, um die gewünschten Resultate zu erzielen?
Zusammengefasst ergibt sich also folgender Anforderungskatalog:

1) Möglichst realistische Grafik
2) Gute Performance, auch auf schwächeren Rechnern
3) Dynamische Licht- und Schattenberechnung
4) Gute Anbindung an 3D-Modellingtools.
5) Leicht zugänglicher und leicht zu bedienender Editor

Zum Zeitpunkt der Recherche für diese Arbeit, gab es zwei Spieleplattformen, die sich zur Verwendung anboten:

1) *Unreal II*, ein 3D-Shooter, mit dazugehörigem Editor *UnrealEd 3.0*
2) *Far Cry*, ein 3D-Shooter aus Deutschland, ebenfalls mit eigenem *Sand-Box-Editor*

1.1.4.1 Unreal II - Die Unreal Engine 2

Unreal II basiert auf der *Unreal Engine 2* und ist ein bekannter Ego-Shooter mit einer breiten Fan-Basis. Zum Erscheinungszeitpunkt war die Grafikengine wegweisend, bot sie doch Features wie hochauflösende Texturen für teilweise nahezu photorealistische Szenen. Der mitgelieferte Editor, mit dem die Entwickler des Spiels selbst die Spielwelt entworfen haben, erlaubt bis in kleinste Detail den Entwurf eigener Welten. Der Preis, den man als Benutzer dafür zahlen muss, ist die relativ lange Einarbeitungszeit in die Bedienung des Editors aufgrund der hohen Komplexität. Dafür ist eine Anbindung an 3ds Max gewährleistet. Um zu prüfen, ob die Engine auch den restlichen Anforderungen des oben aufgestellten Katalogs entspricht, hier ein kurzer Auszug aus der Merkmals-Liste der Engine:

> **Lighting**
> - Per-vertex, Lightmapping:
> - Static lighting through the use of vertex coloring and light maps for high resolution lighting without the runtime calculation overhead.
> - Multiple light types are supported: directional, point lights, and spot lights.
> - Numerous light effects (blinking, wavering, etc.) for dynamic lights.

Shadows
- Shadow Mapping: *Texturing* Basic, Multi-texturing, Projected:
- Projected textures allow the simulation of complex dynamic lighting and shadowing effects on all surfaces, including player shadows and flashlights.
- Many operations supported, such as panning, alpha blending, modulation and more.
- Easy to use mechanism for scrolling, rotating, and scaling textures via artist settings.
- All textures support up to twelve levels of mip-mapping, including mipping down to a solid color.
- Texture animation sequences with variable playback rate. [13]

Wie man an den zwei aufgeführten Punkten erkennen kann, bietet diese Engine nicht die Fähigkeiten, die für diese Arbeit benötigt werden. Dazu ein kurzer Exkurs in die verschiedenen Beleuchtungstechniken, die in Computerspielen eingesetzt werden. Man unterscheidet hier (u.a.) zwei grundlegende Techniken:

1) Lightmapping
2) Dynamische Beleuchtung

Eine *Lightmap* ist eine vorberechnete Textur, in der die Helligkeitswerte für jedes einzelne Pixel in Abhängigkeit von vorher platzierten Lichtquellen gespeichert ist. Diese Textur wird zusätzlich zu der „normalen" Textur auf das Modell aufgebracht und erweckt so den Anschein einer realistischen Beleuchtung. Nachteil dieser Technik ist, dass eine Beleuchtung mit *Lightmaps* immer statisch ist. Das bedeutet, das Modell würde selbst dann noch beleuchtet wirken, wenn die Lichtquelle im Spiel entfernt würde.

Im Unterschied dazu ermöglichen es immer leistungsfähigere Grafikkarten, *dynamische Beleuchtung* durch Verwendung von *Pixelshadern* zur Laufzeit zu berechnen und darzustellen. In Abhängigkeit von der Position der Lichtquelle wird für jedes Pixel einer Textur der jeweilige Helligkeitswert berechnet. Auf diese Weise können bewegliche Lichtquellen realisiert werden, die, in Zusammenhang mit dynamisch berechneten Schatten, der dargestellten Szene eine noch größere Realitätsnähe verleihen.
Es zeigt sich also, dass die Unreal Engine 2 nicht allen der oben formulierten Anforderungen entspricht und sich deshalb für diese Arbeit nicht eignet.

[13] Quelle: 04.02.05, http://www.devmaster.net/engines/engine_details.php?id=39.

1.1.4.2 FarCry - Die CryEngine

Der Ego-Shooter *Far Cry* des deutschen Entwicklerstudios *Crytek* ist etwas neueren Datums als *Unreal II*. Das zeigt sich auch in der Feature-Liste der *CryEngine*. Basierte die *Unreal Engine 2* noch überwiegend auf DirectX 7, kann die *CryEngine* bereits auf die Fähigkeiten der letzten DirectX8/9-fähigen Grafikkartengeneration und damit auf *hardwarebeschleunigte Pixelshader* zurückgreifen. Auch hier also ein Auszug aus den technischen Fähigkeiten der Engine:

> **CryENGINE Sandbox**: Is a real-time game editor offering "What you see is what you
> PLAY" feedback.
> **Renderer**: integrates indoor and outdoor technology seamlessly. Offers rendering
> support for OpenGL & DirectX 8/9, Xbox using latest HW features, PS2 and GameCube.
> **Physics System**: supports character inverse kinematics, vehicles, rigid bodies, liquid, rag doll, cloth and soft body effects. The system is integrated with the game and tools.
> […]
> **Shaders**: A script system used to combine textures in different ways to produce visual
> effects. Supports real time per-pixel lighting, bumpy reflections, refractions,
> volumetric glow effects, animated textures, transparent computer displays, windows,
> bullet holes, and shinny surfaces.
> […]
> **Lighting and Shadows**: A combination of precalculated, real time shadows, stencil
> shadows and lightmaps to produce a dynamic environment. Includes
> high-resolution, correct perspective, and volumetric smooth-shadow implementations
> for dramatic and realistic indoor shadowing. Supports advanced particles technology and any kind of volumetric lighting effects on particles.
> […]
> **Tools Integration**: Objects and buildings created using 3ds max™ or Maya® are
> integrated within the game and editor.
> […]
> **Scripting system**: Based on the popular LUA language. This easy to use system
> allows the setup and tweaking of weapons/game parameters, playing of
> sounds and loading of graphics without touching the C++ code.
> **Modularity**: Entirely written in modular C++, with comments, documentation and
> subdivisions into multiple DLLs.[14]

Zusammenfassend lässt sich Folgendes festhalten: Die *CryEngine* bietet einen intuitiv bedienbaren WYSIWYG-Editor, der schon im Editor die spätere Spielgrafik darstellt.

[14] Quelle: 04.02.2005, Crytek Webseite: http://www.crytek.de/technology/index.php?sx=cryengine (Webseite liegt dieser Arbeit im .pdf-Format bei.).

Die Engine unterstützt und nutzt die *Shader*-Fähigkeiten aktueller Grafikhardware und bietet so pixelgenaue Beleuchtung in Echtzeit.[15]
Darüber hinaus unterstützt die *CryEngine* neben anderen Beleuchtungstechniken auch die Berechnung von Echtzeitschatten.
Modelle können in *3ds Max* oder in *Maya* erstellt werden und per mit geliefertem Plugin für den *Sandbox-Editor* exportiert werden.
Die *CryEngine* erfüllt also alle Anforderungen des oben formulierten Katalogs.
Somit wäre die Wahl der Werkzeuge abgeschlossen: *3ds Max* für die Erstellung des Höhlenmodells, in Kombination mit der Integration des Modells in die *CryEngine* über den relativ leicht zugänglichen *Sandbox-Editor*, welcher die gewünschte Interaktivität gewährleistet.

2 Erstellung des Höhlenmodells mit 3ds Max 6

Die Rekonstruktion der Höhle von Altamira und die damit verbundenen Untersuchungen sind exemplarisch zu verstehen und dienen in erster Linie dazu, die Möglichkeiten und eventuell auch die Einschränkungen aufzuzeigen, die die gewählte Software-Kombination für wissenschaftliche Zwecke bietet.
Es sind durchaus auch andere Projekte denkbar, die sich mit den gewählten Mitteln realisieren lassen würden, wie z.B. die Erstellung eines Modells des antiken Pompeji, möglicherweise für ein Museum, das es Besuchern ermöglichen würde, interaktiv die Straßen der Stadt zu durchstreifen, bevor sie zerstört wurde.
Bei der Erstellung des Modells gibt es keine fest vorgeschriebene Vorgehensweise, da sich die *richtige* Art und Weise von Fall zu Fall unterscheidet.
Hilfreich sind in jedem Fall möglichst genaue Angaben über die jeweilige Geometrie des zu modellierenden Objekts.
Der Versuch, genauere Geometriedaten über die Altamirahöhle zu bekommen, war also der erste Schritt. Zwar existieren diese, da z.B. im *Deutschen Museum* in München zumindest in Teilen ein Nachbau der Höhle ausgestellt ist, leider war aber in der Kürze der Zeit nicht zu klären, wer diese Daten besitzt und wo die Rechte daran liegen. Ähnliches gilt für das *Museum in Altamira* selbst. Auch dort existiert ein Nachbau der Höhle, genauere Daten waren aber auch hier nicht erhältlich.

2.1 Vom Grundriss zum Modell

Zur Verfügung stand ein Grundriss der Höhle (s.Abb.1) mit eingezeichneten Höhenlinien, die die Unebenheiten und Gefälle des Bodens nachzeichnen. Dieser liefert erste Anhaltspunkte über die Größenverhältnisse und die horizontale Ausdehnung der Höhle. Um diese am PC verwertbar zu machen, wurde der Grundriss zunächst einges-

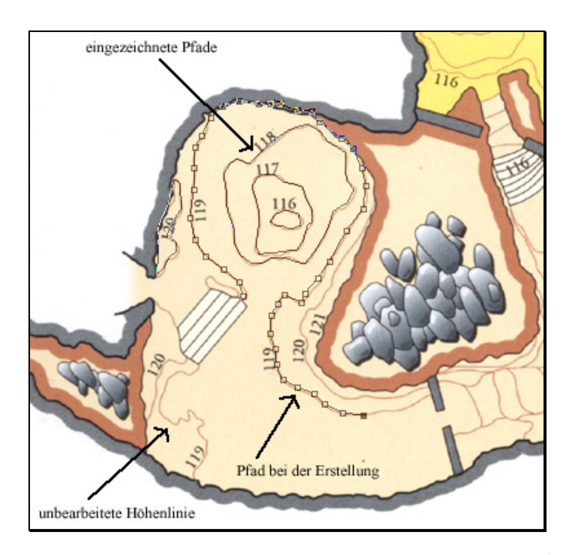
[15] Vgl. Kapitel 1.4.1, Exkurs zu *dynamischer Beleuchtung*.

cannt. Nachgebaut wurde der für die Problemstellung relevante Teil der Höhle: Der Eingangsbereich und natürlich der *Saal der polychromen Decke*.



Abbildung 3: der nachzubauende Teil der Höhle

In *Adobe Photoshop* wurden anschließend die äußeren Umrisse und die Höhenlinien manuell mit sog. *Pfaden* nachgezeichnet. Diese Pfade lassen sich in *Photoshop* als *Illustratorpfade (als .ai-Datei)* exportieren.



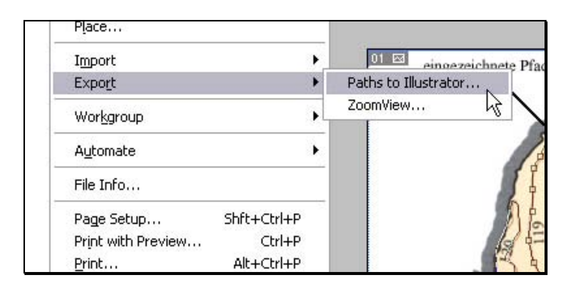




Abbildung 4: Nachzeichnen der Höhenlinien mit Pfaden **Abbildung 5: Export der Pfade in Photoshop**

Die so exportierten Pfade lassen sich in 3ds Max importieren, so dass man einen Grundriss der Höhle erhält.

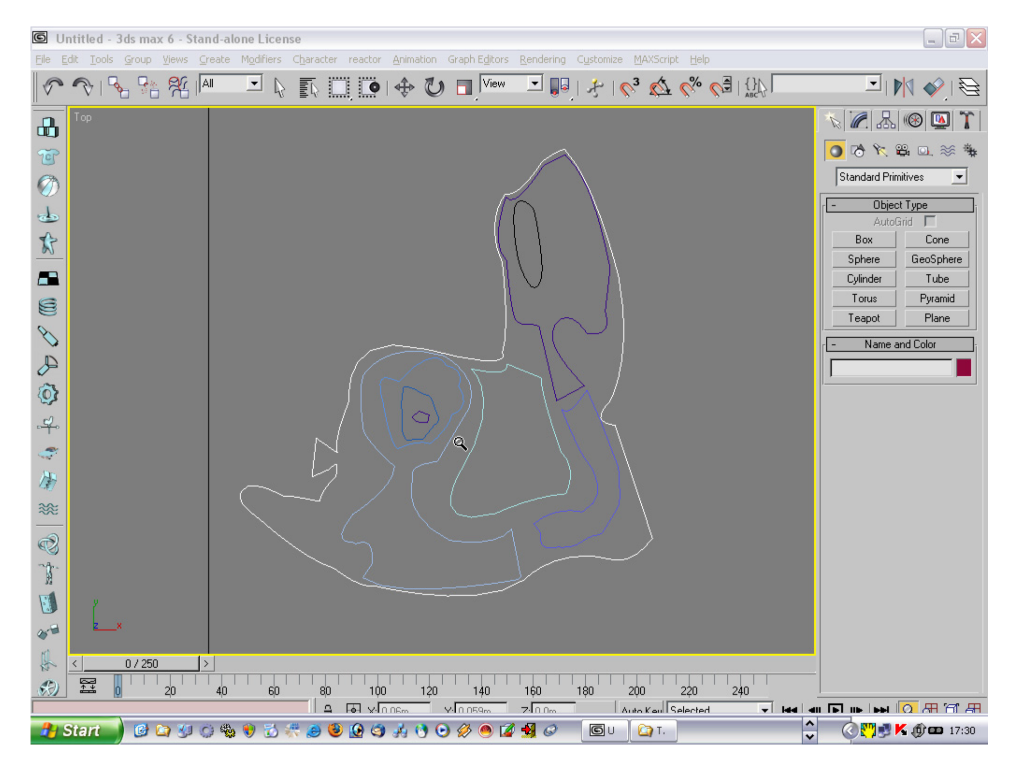


Abbildung 6: Importierte Pfade in 3ds Max

Jeder einzelne, importierte Pfad (hier verschieden gefärbt) wird nun als separates Objekt in *3ds Max* verwaltet. So kann man die Pfade, entsprechend ihrer Höhenangabe verschieben, und erhält aus einem zweidimensionalen Grundriss ein dreidimensionales Bodenmodell, repräsentiert durch Höhenlinien.

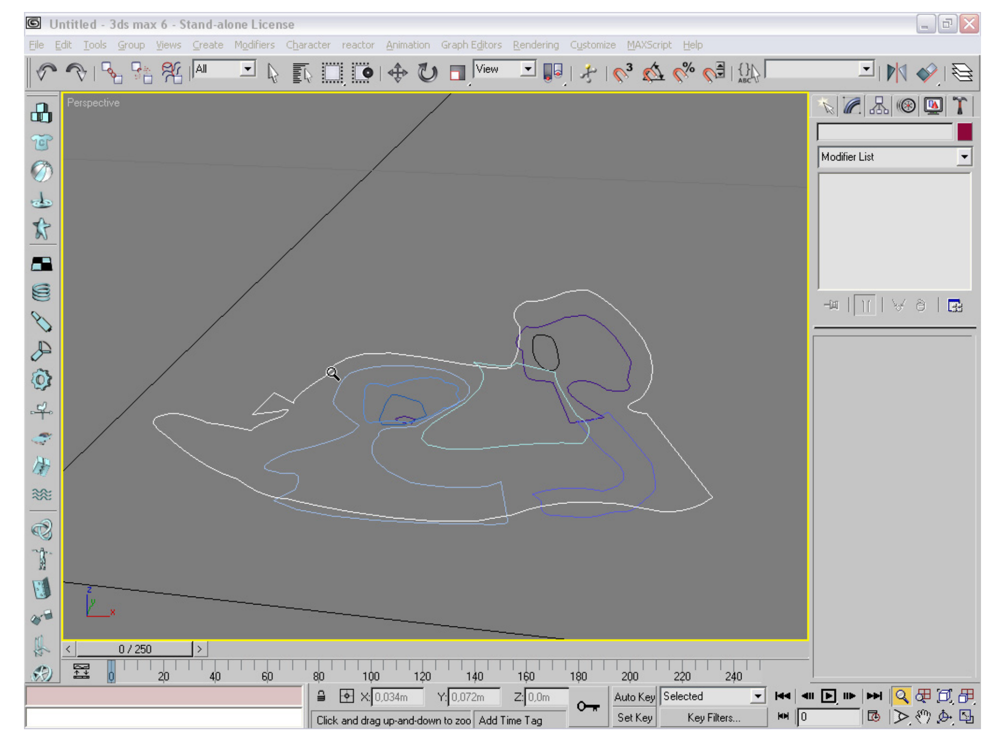


Abbildung 7: Höhenverschobene Pfade, Perspektivische Ansicht

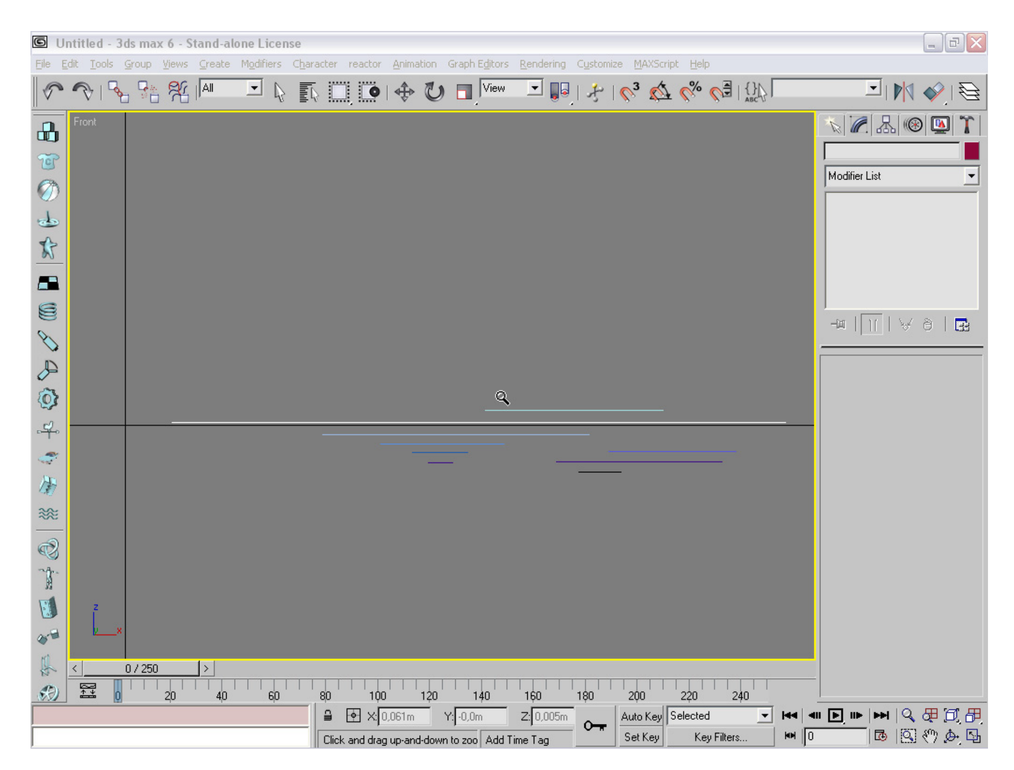


Abbildung 8: Höhenverschobene Pfade, Seiten-Ansicht

3ds Max ist nun in der Lage, aus diesen Höhenlinien ein *Terrain*-Modell zu erzeugen, in dem es den Bodenverlauf zwischen den Linien berechnet und die nötigen Polygone einfügt, um eine Oberfläche zu erzeugen.

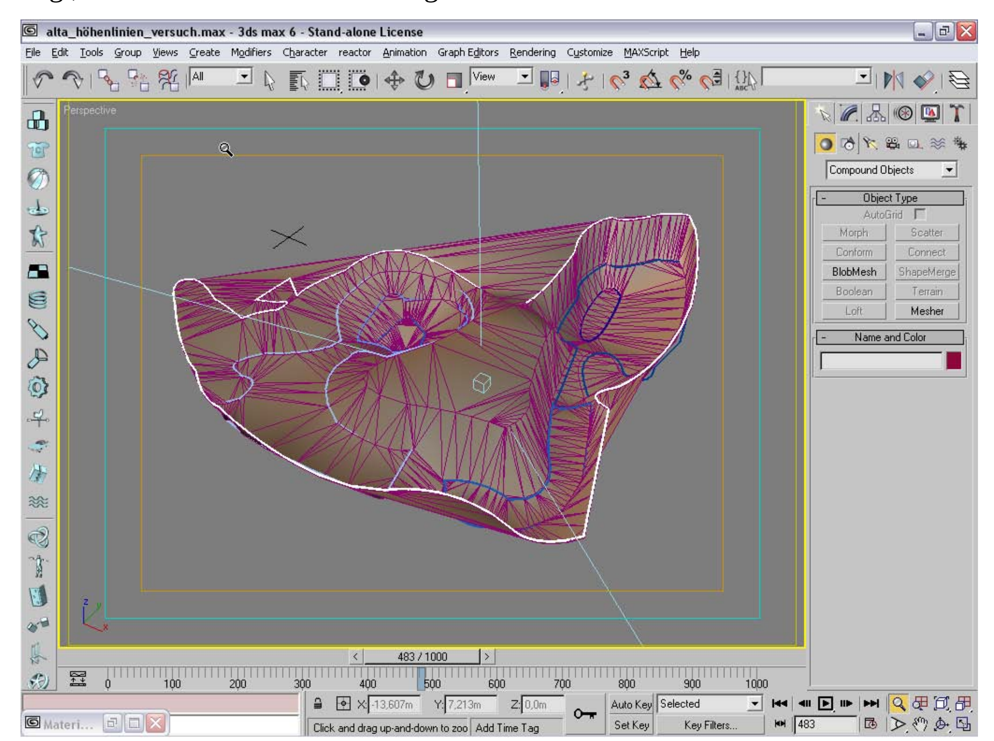


Abbildung 9: Das von 3ds Max erstellte Bodenmodell

Es entsteht eine Art „Gipsabdruck“ des Bodens der Höhle. Da, wie schon erwähnt, keine weiteren Geometriedaten zur Verfügung stehen, soll dieses Bodenmodell den Ausgangspunkt für ein Höhlenmodell bilden, das zumindest in Größe und Grundriss mit der tatsächlichen Höhle übereinstimmt. Die Lage der Decke und der Wände konn-

te nur aus Photographien erschlossen werden. Im Hinblick auf die Zielsetzung dieser Arbeit, in der die Altamirahöhle ohnehin nur exemplarischen Charakter hat, wurde dies in Kauf genommen.

Da dem Bodenmodell nun noch Wände und eine Decke fehlten, mussten diese in *3ds Max* durch Einfügen weiterer Polygone erzeugt werden. Wie sich während der weiteren Bearbeitung herausstellte, ordnet das *Terrain-Modul* die Polygone ungünstig an. So wurde die Erstellung eines geordneten *Drahtgittermodell*s unnötig erschwert. Deshalb wurde das erstellte Bodenmodell als Referenz für ein neues, geordneteres Modell verwendet. Der Unterschied im Aufbau wird in den nächsten Abbildungen deutlich.

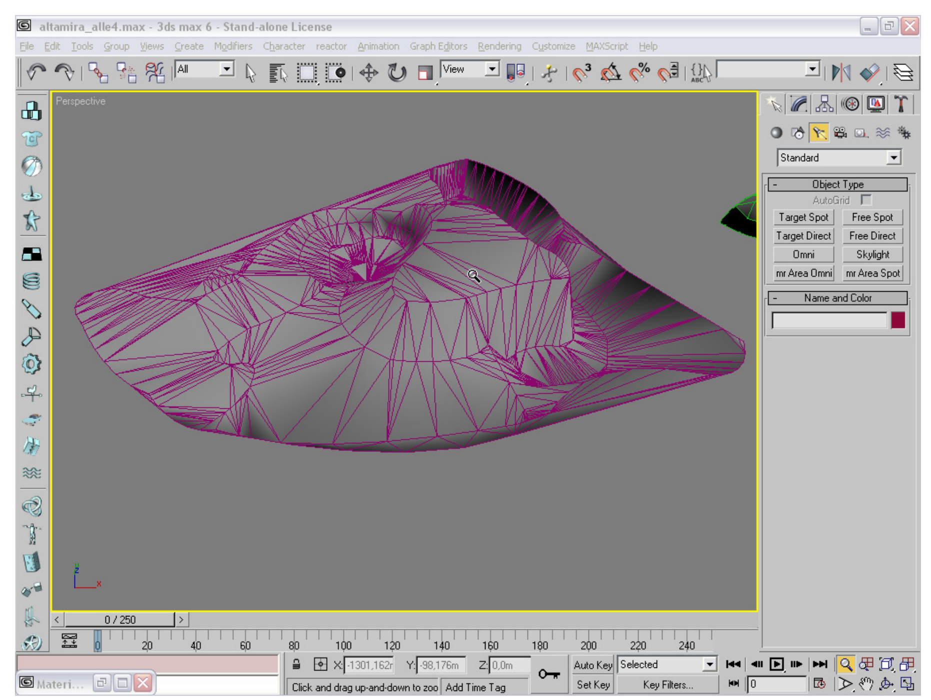


Abbildung 10: Von 3ds Max erzeugtes Bodenmodell

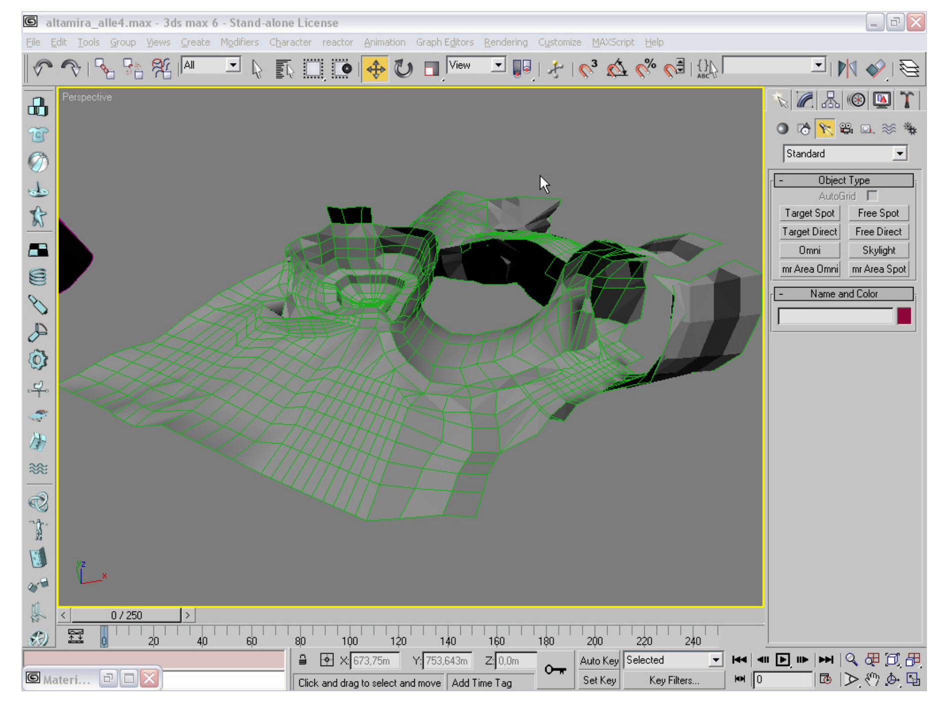


Abbildung 11: Neues, geordneteres Drahtgittermodell

Ausgehend von diesem neuen Modell wurden weitere Polygone hinzugefügt und ausgerichtet, so dass sich schließlich ein geschlossener Raum mit Wänden und Decke ergab.

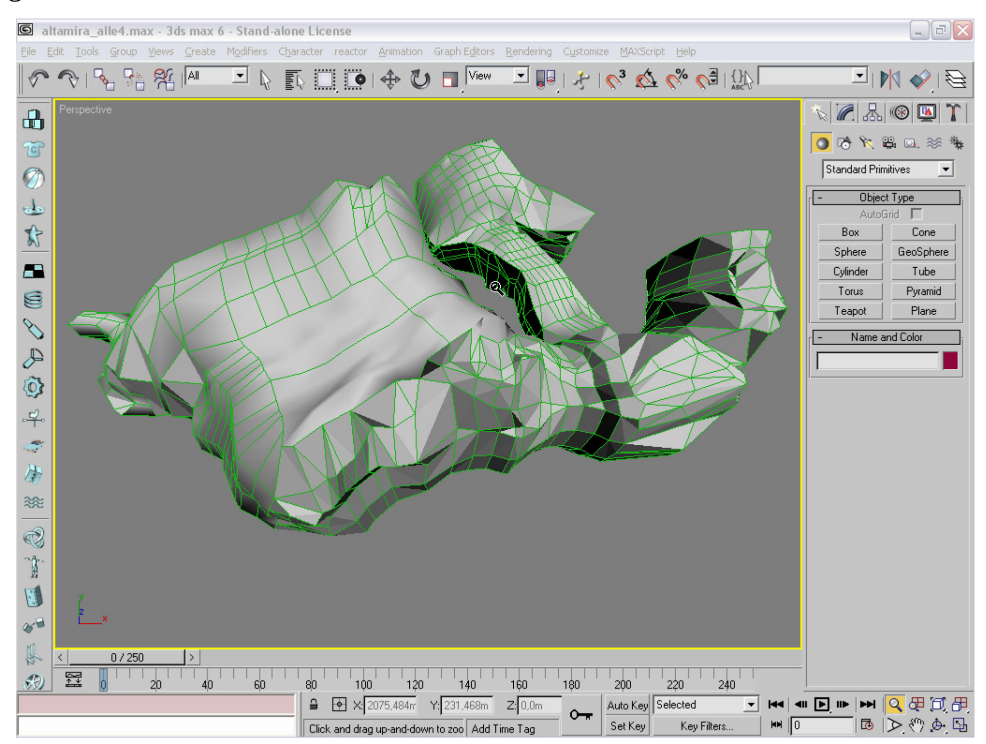


Abbildung 12: Geschlossenes Höhlenmodell

Das Modell der Höhle besteht zu diesem Zeitpunkt aus relativ wenigen Polygonen und wird im weiteren Verlauf verfeinert. Dazu gehört die gesonderte Erstellung der polychromen Decke.

2.2 Die Erstellung der polychromen Decke

Da die Decke des polychromen Saales von besonderer Bedeutung ist, wurde sie einzeln erstellt und später mit der restlichen Höhle verbunden. Auf diese Weise konnte die Decke wesentlich detaillierter ausgeführt werden, ohne die Gesamtpolygonzahl zu sehr in die Höhe schnellen zu lassen. Während die restliche Höhle aus ca. 7000 Polygonen besteht, setzt sich allein die Decke aus ca. 8500 Polygonen zusammen.

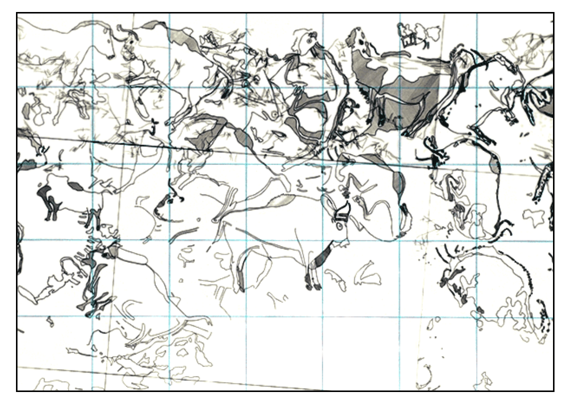
Als erster Schritt musste zunächst die Deckentextur erstellt werden. Im Folgenden wird kurz die Vorgehensweise erläutert.

Um einen Anhaltspunkt über die Lage der einzelnen Zeichnungen und Malereien an der polychromen Decke zu bekommen, wurde die Abzeichnung der Decke aus Abb.2 herangezogen. Dann begann in *Photoshop* die puzzleartige Zusammensetzung der Deckentextur.[16] Auf Grundlage der vorliegenden Abzeichnung der polychromen De-

---

[16] Die Photos, die hier Verwendung fanden, wurden mit freundlicher Genehmigung von Dr. Andreas Pastoors vom Institut für Ur- und Frühgeschichte der Universität Köln zur Verfügung gestellt. Dr. Pastoors ist Mit-Herausgeber der CD-Rom „Bilder im Dunkeln – Höhlenkunst der

cke wurden nun die einzelnen Photos, soweit vorhanden, zu einer Gesamtansicht der Decke zusammengesetzt.



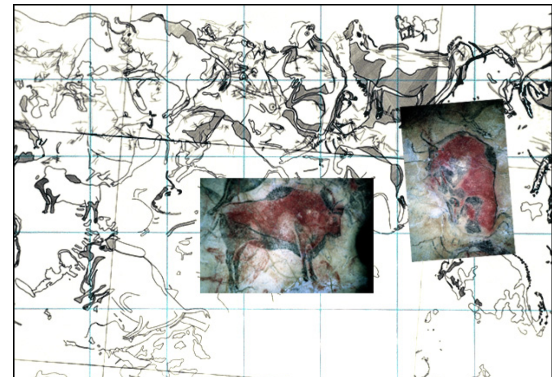




Abbildung 13: Konstruktion der Deckentextur

Nachdem die einzelnen Photos angepasst und platziert waren, wurde die Beschaffenheit der gesamten Decke angepasst, indem Ränder und Schattierungen soweit möglich entfernt wurden (siehe letzte Abbildung).

Nun musste noch das 3D-Modell der Decke konstruiert werden. Ausgehend von einer rechteckigen glatten Fläche wurde ein *Rauschen* eingefügt, um eine unregelmäßige Oberfläche zu erzeugen.

Eiszeit“, die 1150 Bilder des Bühnenbildners und Photographen Heinrich Wendel enthält. Die Sammlung Wendel gehört zu den größten Bildarchiven eiszeitlicher Höhlenkunst weltweit und enthält Photographien aus 45 Bilderhöhlen Südfrankreichs und Nordspaniens, darunter auch die Höhle von Altamira.

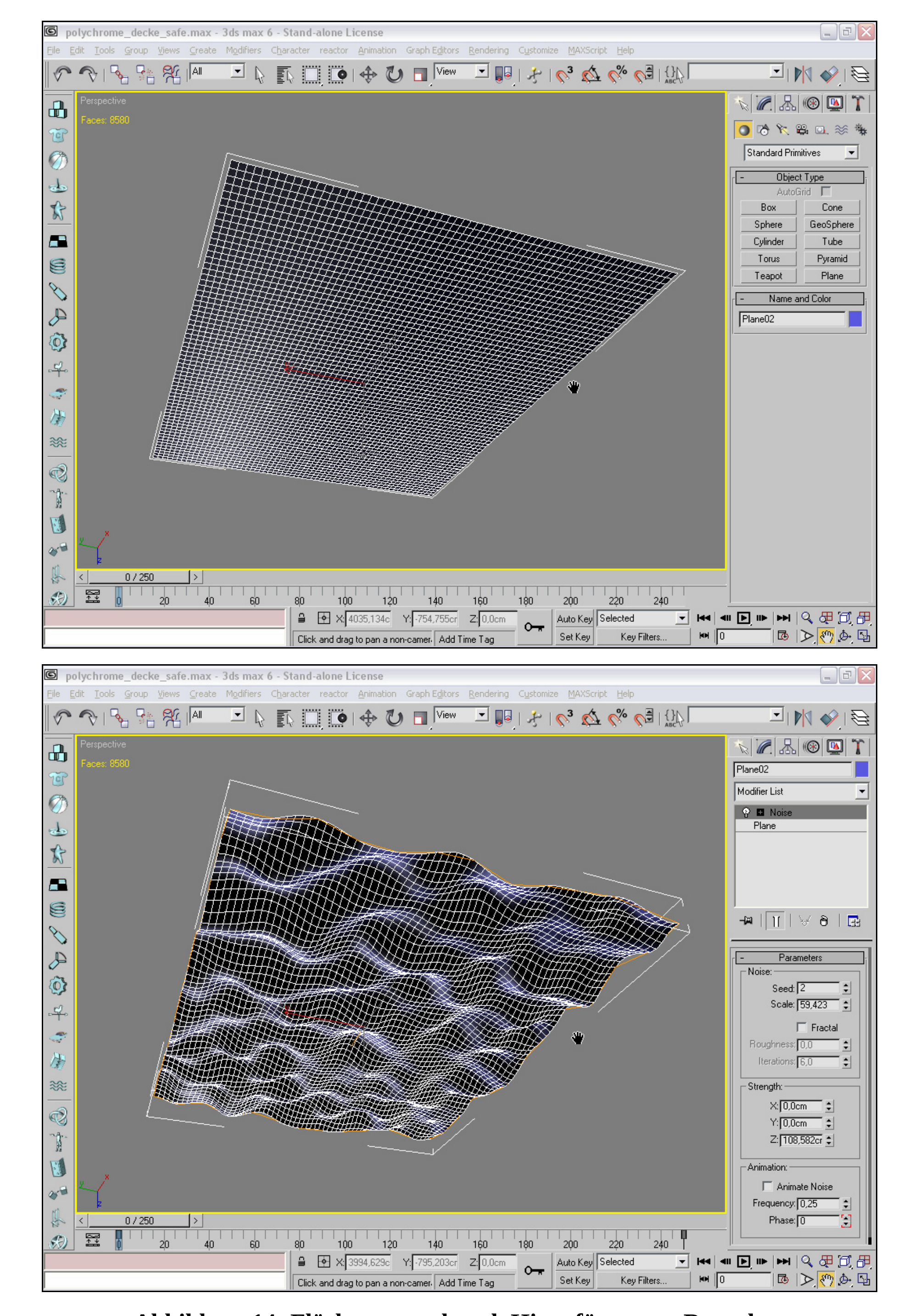


Abbildung 14: Fläche vor und nach Hinzufügen von Rauschen

Nun kann die Textur aufgebracht und die Decke an einigen charakteristischen Punkten daran angepasst werden. Die wichtigste Stelle ist hier die Gruppierung der drei liegenden Wisente. Jeder dieser Wisente ist an eine natürliche Erhebung der Decke angepasst, so dass sie sehr plastisch wirken. Dies illustrieren die folgenden Photographien der realen Decke:





Abbildung 15: Photo der realen Decke (Markierung zeigt die drei Wisente) und Einzelansicht (rechts)

Nachdem die Textur aufgebracht worden ist, können die Erhebungen herausmodelliert werden, um den Gesamteindruck der realen Decke möglichst detailliert zu erhalten.

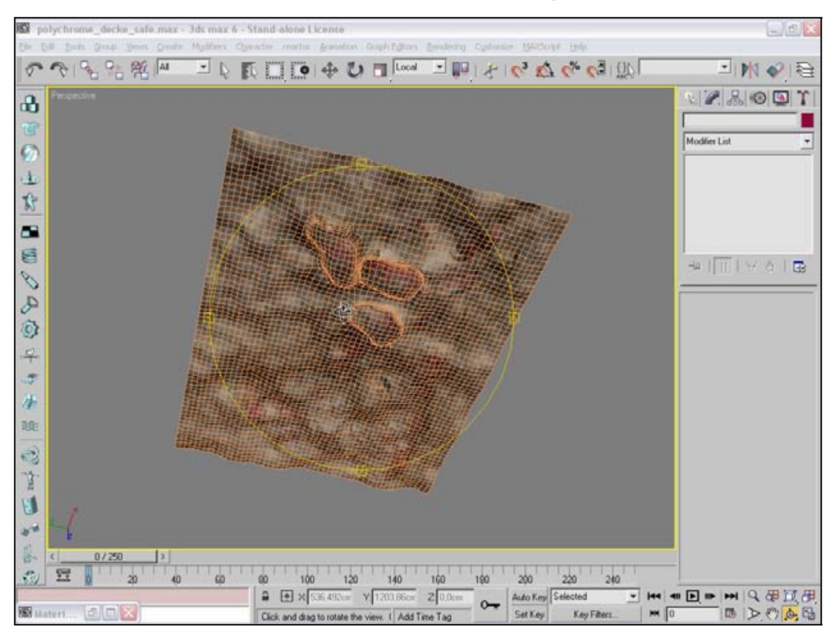


Abbildung 16: Die Fläche mit den herausmodellierten Erhebungen für die Wisente

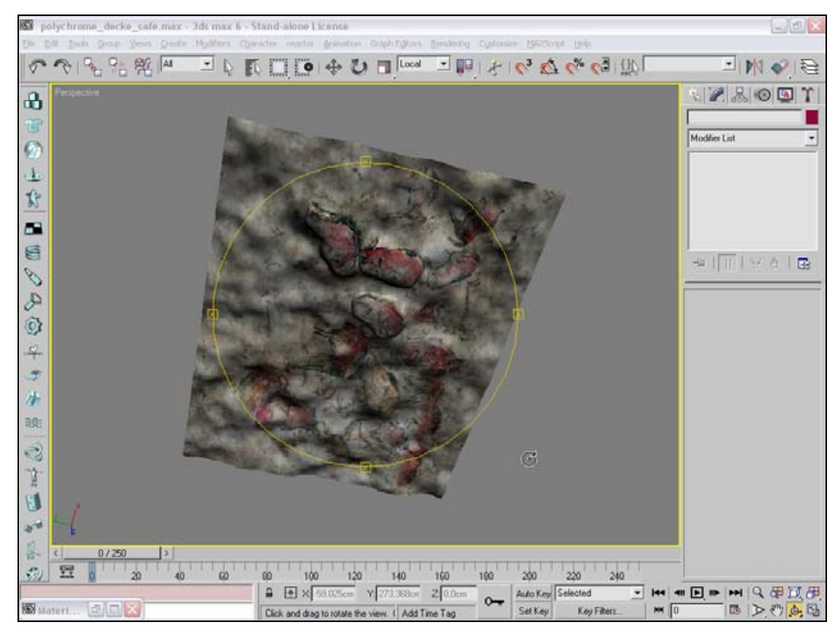


Abbildung 17: Die fertige Decke

2.3 Verbinden von Decke und Höhle

Nachdem die Textur aufgebracht, das 3D-Modell der Decke erstellt und an die Textur angepasst worden ist, muss die Decke noch mit der restlichen Höhle verbunden werden.

Dazu werden die bereits vorhandenen Polygone der Decke des polychromen Saales entfernt, um dann die neu erstellte, detailliertere Decke wie einen Deckel auf den polychromen Saal zu setzen.

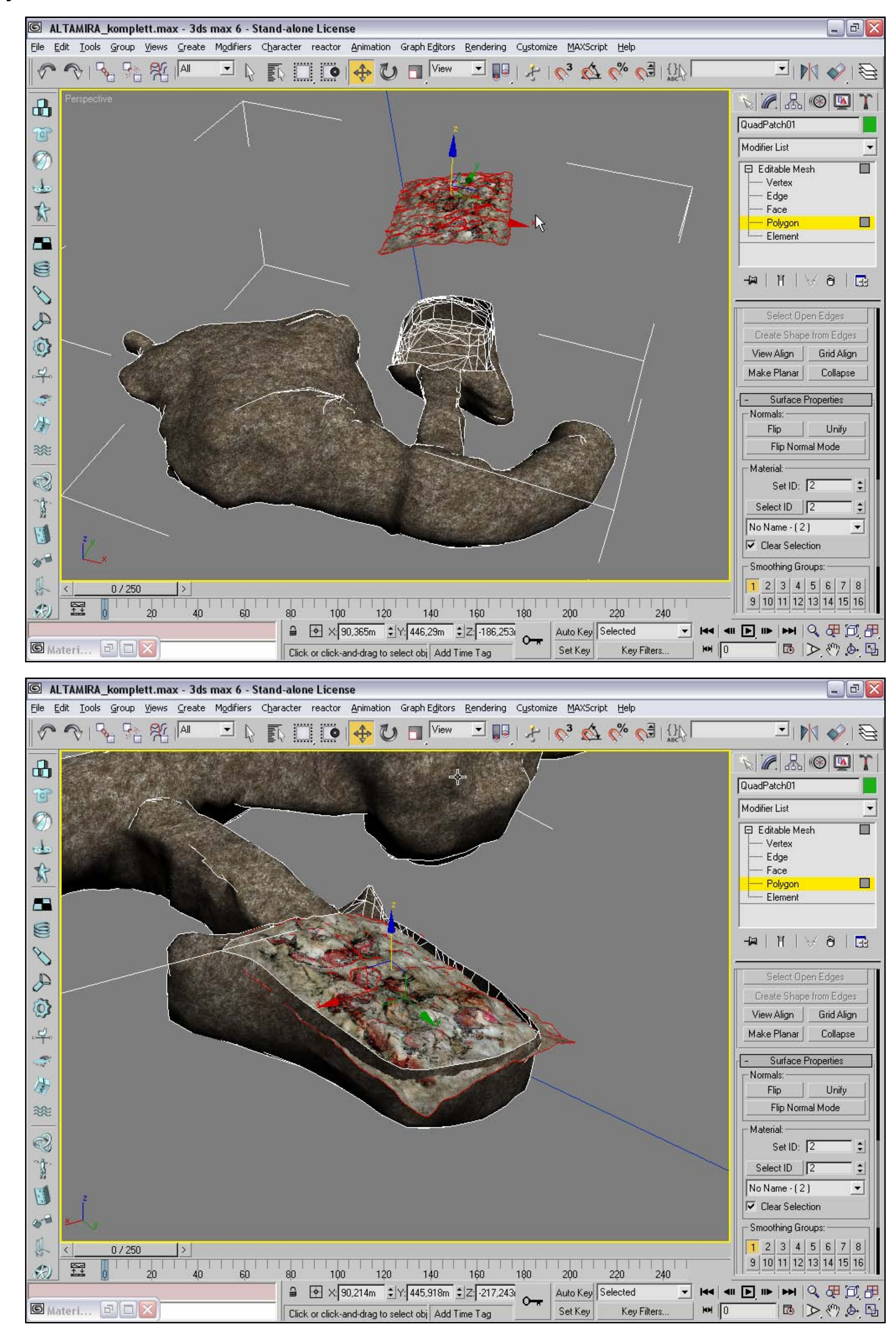


Abbildung 18: Aufsetzen der texturierten Decke

Dass die Decke an den äußeren Rändern übersteht, kann vernachlässigt werden, da die Höhle später nur noch von Innen zu sehen sein wird. Nach den wenigen, zur Verfügung stehenden Aufzeichnungen, ist die Decke im polychromen Saal an den höchsten Stellen ca. 2m hoch[17], also etwas über Kopfhöhe. Demnach wurde die Decke des Modells in etwa in die gleiche Position gebracht. Für spätere Versuche es ist aber auch nachträglich möglich, die Position der Decke ohne großen Aufwand zu verändern.

2.4 Texturierung der Höhle und Export für den Sandbox-Editor

Um die restliche Höhle noch realistischer aussehen zu lassen, wurde das Drahtgittermodell durch Einfügen zusätzlicher Polygone verfeinert. Dies geschieht automatisch mittels eines in *3ds Max* enthaltenen Werkzeugs, des sog. *Mesh-Smooth*-Modifiers. Dieser unterteilt die vorhandenen Polygone in kleinere, was insgesamt zu einem glatteren Erscheinungsbild führt. Vor Anwendung des Modifiers bestand die Höhle aus ca. 3700, danach aus ca. 7000 Polygonen.

Anschließend wurde noch der Rest der Höhle mit einer einfachen Felstextur ausgestattet, um ein realistischeres Erscheinungsbild zu erzielen.

Als letzter Feinschliff wurden die Texturen, sowohl die der Decke, als auch die Felstextur der Wände, mit einer zusätzlichen *Bump-Map* versehen[18]. Eine solche *Bump-* oder auch *Relief-Map* lässt

> „[…] eine Oberfläche scheinbar uneben aussehen. An den hellen Stellen der projizierten Bilddatei tritt die Oberfläche etwas hervor. Somit lassen sich sehr einfach Strukturen oder Gravuren, Steinfugen oder Landschaftsstrukturen erzeugen. Hier haben die Farben und Graustufen der Bilddatei keinen Einfluss auf die Objektfarbe, sondern nur auf die Oberflächenstruktur. Diese Relief-Maps sind eine gute Methode, Objekte nicht nur bemalt, sondern wirklich realistisch darzustellen, ohne die Flächenzahl um das Vielfache zu erhöhen."[19]

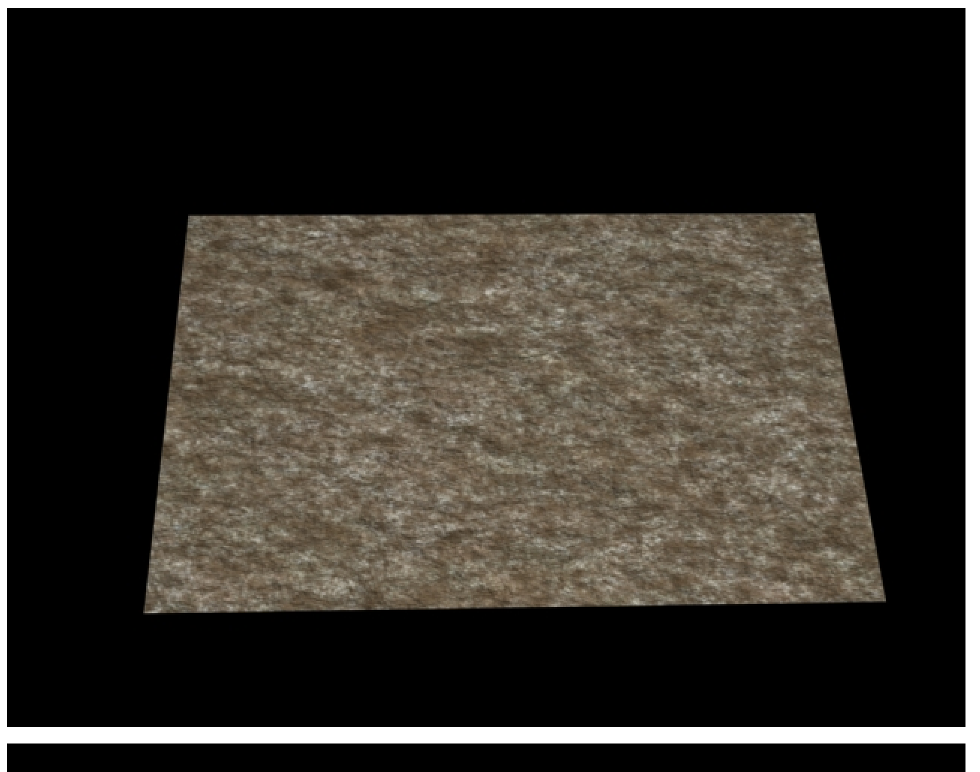
Die nächsten zwei Abbildungen machen den Effekt etwas anschaulicher:

---

[17] Vgl. Antonio Beltrán, Altamira, 1998. S.65, 3. Absatz.

[18] Als Bump-Map wurde in beiden Fällen jeweils die gleiche Bilddatei verwendet, wie für die Textur selbst.

[19] Vgl. Immler, Christian: 3ds max 5, 2003. S. 158 f.



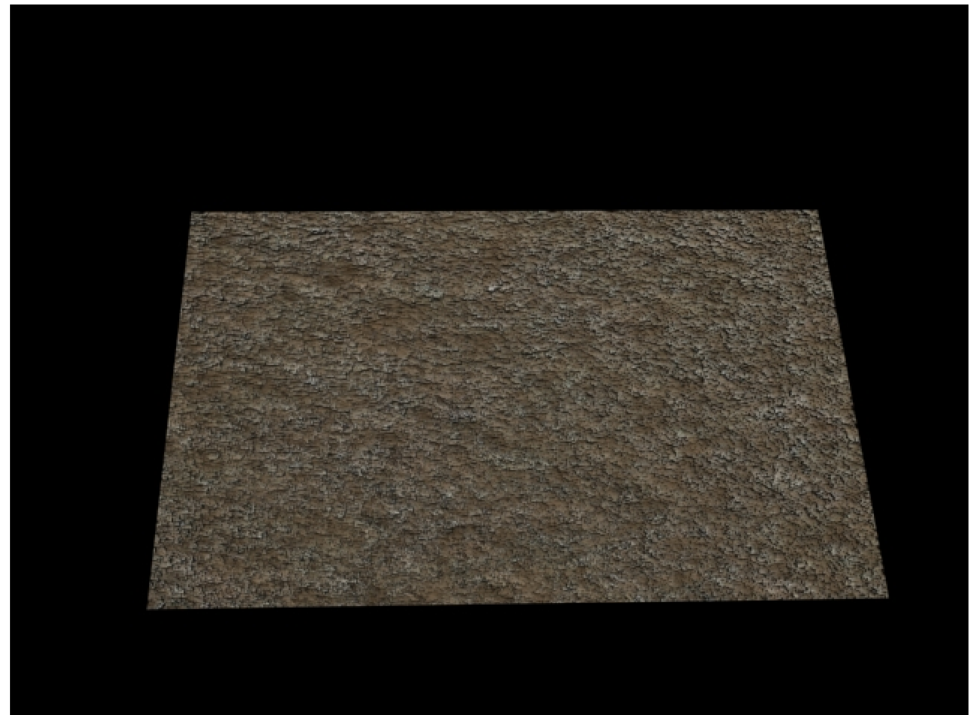


Abbildung 19: Verwendete Fels-Textur, ohne (oben) und mit (unten) Bump-Map

Nachdem Fertigstellung des Höhlenmodells, muss dieses noch in das proprietäre Format des *Sandbox-Editors* der *CryEngine* exportiert werden. Im Lieferumfang des Editors befindet sich zu diesem Zweck ein Plugin, der *Crytek Exporter*, das direkt in *3ds Max* integriert wird und so einen unkomplizierten Datentransfer zwischen den beiden Applikationen ermöglicht.

Zuvor ist aber - für eine korrekte Darstellung des Modells im *Sandbox-Editor* - sicherzustellen, dass einige wichtige Punkte beachtet werden. So ist es erforderlich für die verwendeten Texturen einen der mitgelieferten *Crytek-Shader* zu verwenden, da

das Modell sonst später im Editor nicht sichtbar ist. Außerdem ist sicherzustellen, dass die Flächen-Normalen der Texturen in die richtige Richtung zeigen. Hierzu eine kurze Erklärung.

Die Flächennormalen sind gedachte Vektoren, die wie ein Lot auf der texturierten Oberfläche stehen. Sie zeigen an, in welche Richtung eine Textur zeigt. Da später das *Innere* der Höhle untersucht werden soll, müssen die Normalen, die per Default alle nach *Außen* zeigen, nach Innen gedreht werden, da die Höhlenwände und der Fußboden sonst später im Editor keine Substanz haben. Gleiches gilt für die Decke, da sonst später z.B. der Lichteinfall nicht korrekt berechnet wird.

Jetzt ist die Höhle fertig für den Export. Im weiteren Verlauf soll nun dargestellt werden, wie das Modell in die *CryEngine* integriert werden kann und welche Möglichkeiten sich hinsichtlich der ursprünglichen Beleuchtungshypothese bieten.

3 Der CryEngine Sandbox-Editor

Der *Sandbox-Editor* ist Bestandteil des *Crytek CryEngine SDK 1.1*.[20] ebenfalls enthalten ist der *Crytek Exporter* für *3ds Max* und *Maya*. Die Bedienoberfläche des Editors ist stark an *3ds Max* angelehnt, so dass man sich nach der Einarbeitung in *3ds Max* relativ rasch und intuitiv zurechtfindet.[21]

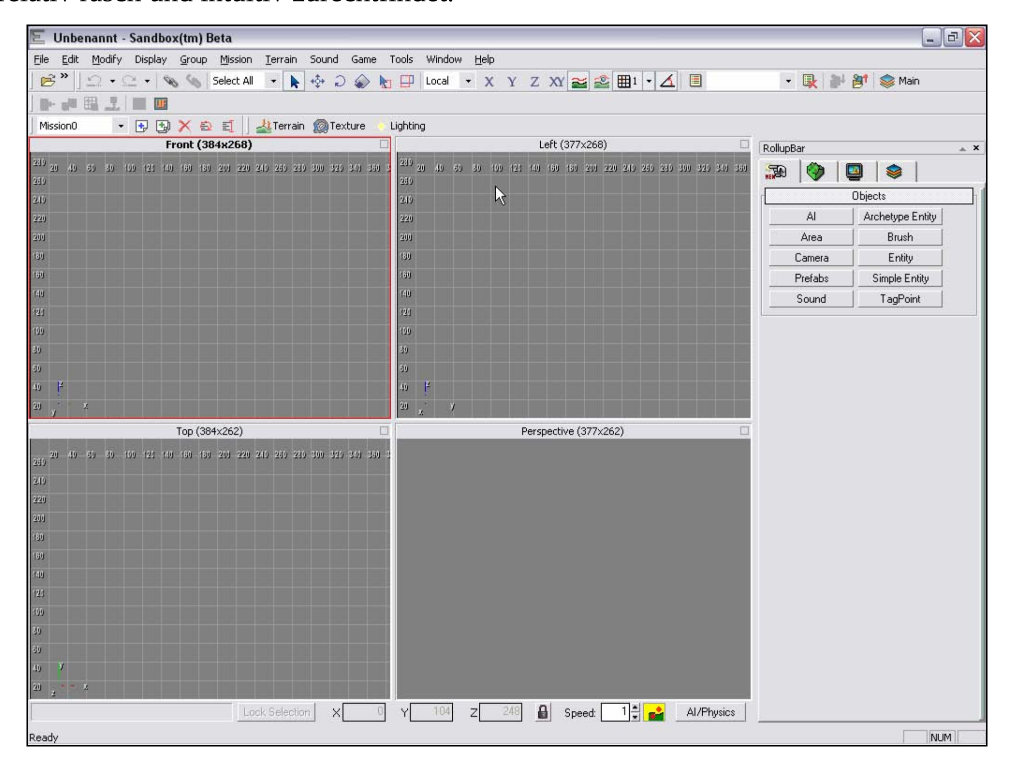


Abbildung 20: Bedienoberfläche des Sandbox-Editors

---

[20] Damit Der Editor und das SDK lauffähig sind, muss das Spiel FarCry installiert sein, da der Editor auf alle Modelle und Ressourcen des Spiels zugreift.

[21] Das Benutzerhandbuch zum Editor liegt in der Version 1.1 als pdf-Datei der Arbeit bei.

Da die Entwickler die Levels des Spiels ebenfalls mit der Sandbox erstellt haben, hat man hier ein sehr umfangreiches, mächtiges Werkzeug. Eingebunden in die *CryEngine* und den Editor ist die Scriptsprache LUA, über die viele Manipulationen an Objekten vorgenommen werden können, ohne den Quellcode der Engine zu verändern.
Um eigene Modelle im Editor zu verwenden müssen diese an der richtigen Stelle der vorgegebenen Ordnerstruktur, im richtigen Format, abgelegt werden.[22]
Einer der größten Vorteile des Editors besteht in seiner sog. „What you see is what you play“-Fähigkeit. Das bedeutet, die Darstellung innerhalb des Editors unterscheidet sich nicht von der In-Game-Grafik. Außerdem kann man per Tastendruck (Strg-G) jederzeit fließend in den *Game-Modus* umschalten, so dass man ständig die Kontrolle über seine Arbeit behält.
Warum diese Funktion besonders wichtig ist, wird später noch eingehender erläutert.

3.1 Einbindung des Höhlenmodells in die Game-Engine

Zunächst muss das Höhlenmodell in den Editor importiert werden. Dazu muss es im .cgf-Format vorliegen. Außerdem müssen alle verwendeten Texturen in Bilddateien im .dds-Format[23] umgewandelt werden.
Zur Erinnerung noch einmal kurz die Zielsetzung dieser Arbeit: Es soll anhand eines hinreichend realistischen Höhlenmodells untersucht werden, wie sich die Höhle oder vielmehr die Malereien und Zeichnungen an der polychromen Decke unter verschiedenen Beleuchtungssituationen darstellen. Als eine mögliche Hypothese kann so die eingangs formulierte Frage verfolgt werden, in wie weit die Beleuchtungssituation die Anordnung der Kunstwerke beeinflusst haben könnte.
Für dieses Vorhaben ist die Fähigkeit der *CryEngine*, Licht und Schatten pixelgenau in Echtzeit darzustellen von großer Wichtigkeit.
Als zusätzlichen Bonus erhält man die Möglichkeit, die Höhle in ihrem Urzustand zu untersuchen, d.h. in diesem Fall konkret, vor Aushebung des Fußbodens im Saal der polychromen Decke, die erfolgte, um den Saal, der an einigen Stellen nur ca. 1 m hoch war, besser begehbar zu machen.
Man kann die Höhle also beinahe unter den Bedingungen untersuchen, unter denen sie auch der damalige Künstler vorgefunden hat. Zusätzlich sind „bauliche Veränderungen“ am Modell jederzeit ohne großen Aufwand möglich, so dass das Modell auch in der Zukunft einem neuen Erkenntnisstand schnell angepasst werden könnte.
Ausgangspunkt eines *FarCry*-Levels ist zunächst immer eine „leere Landschaft“, die anfangs nur eine Wasserfläche und die sog. *Skybox* enthält. Die *Skybox* liefert den

---

[22] Für eine genaue Installationsanleitung siehe Kapitel 5: Installationsanleitung.
[23] Hierfür stehen Plugins für Photoshop und 3ds Max zur Verfügung: Quelle Nvidia-Webseite, 08.02.2005: http://developer.nvidia.com/object/nv_texture_tools.html. (Die DDS-Tools liegen der Arbeit au DVD bei.).

Hintergrund des Levels, sowie die Grundbeleuchtung durch die Sonne, als zentrale Lichtquelle. Die folgende Darstellung veranschaulicht dies und vermittelt einen ersten Eindruck der hohen Grafikqualität:

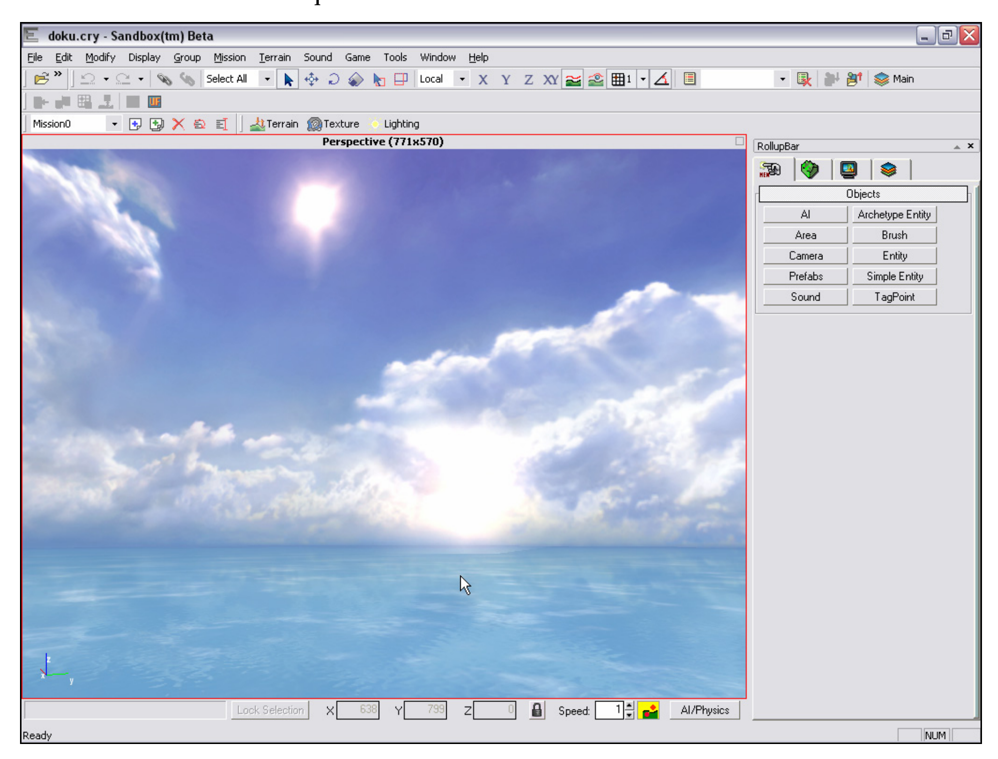


Abbildung 21: Ausgangspunkt der Levelerstellung: Wasser und Skybox mit Sonne

Nun kann das Höhlenmodell eingefügt werden.

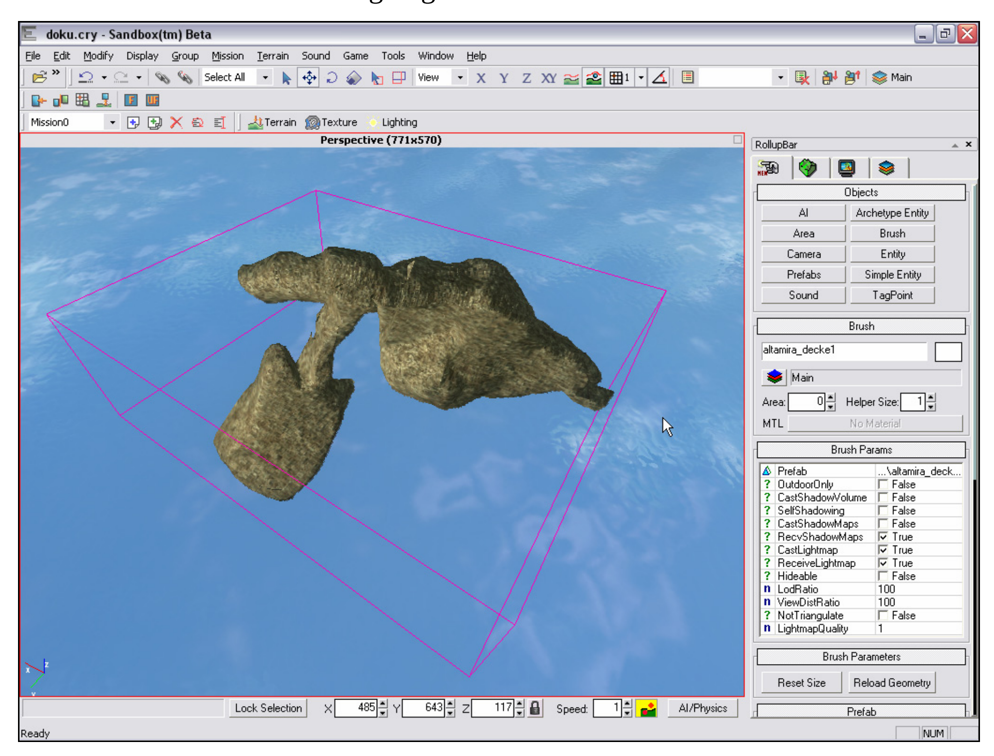


Abbildung 22: Die Höhle im Sandbox-Editor

Dass die Höhle über dem Wasser schwebt, ist später nicht mehr relevant, da man sich ausschließlich im Innern der Höhle bewegt.[24]

Schon zu diesem Zeitpunkt ist es möglich, „In-Game“ mit einer Spielfigur die Höhle zu erkunden:

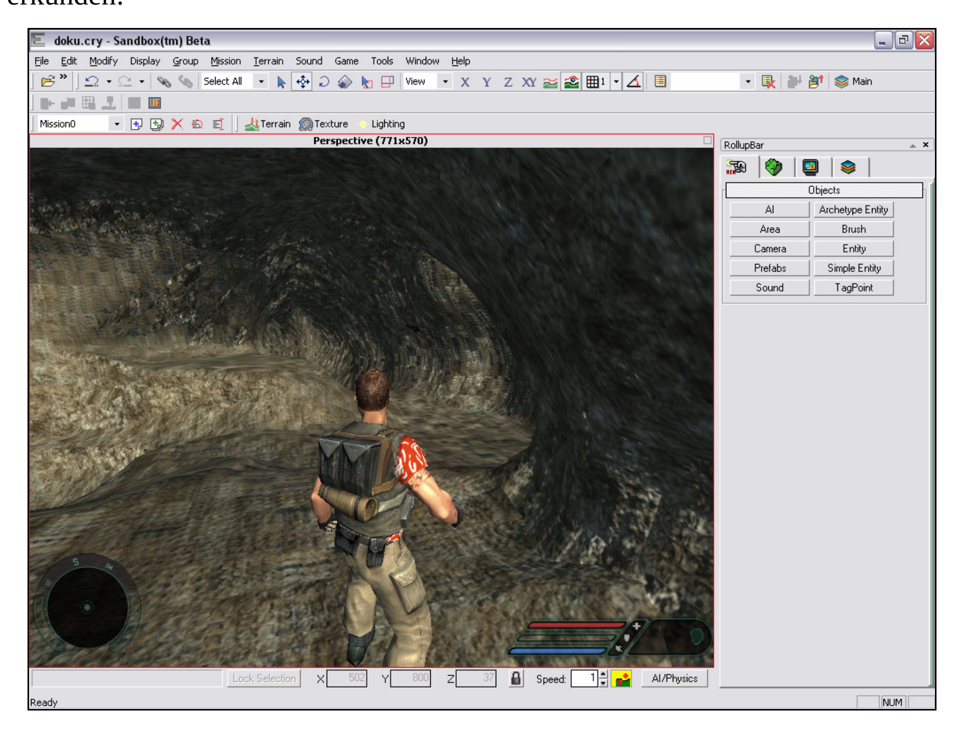


Abbildung 23: Die In-Game Ansicht der Höhle

Bei genauerer Betrachtung wird deutlich, dass das Innere der Höhle bereits beleuchtet ist. Dies resultiert aus der globalen Beleuchtung durch die Sonne und entspricht weder der Realität, noch den Erfordernissen der Zielsetzung dieser Arbeit. Um sicherzustellen, dass die Höhle nicht zusätzlich beleuchtet wird, ist die Errichtung einer sog. *VisArea* erforderlich. *VisArea* werden mit dem Editor mitgeliefert. Sie trennen Innenbereiche von Außenbereichen ab und sorgen dafür, dass geschlossene Räume nur durch die darin platzierten Lichtquellen beleuchtet werden.

[24] Natürlich wäre es auch möglich, die Höhle in eine landschaftliche Umgebung einzubinden. Vorstellbar wäre dies zum Beispiel für ein Museum. Ein Besucher könnte, an einem PC-Terminal, ein Modell der Landschaft rund um die Höhle durchstreifen und dann schließlich die Höhle erkunden. Für das Thema dieser Arbeit reicht die „schwebende“ Höhle allerdings aus.

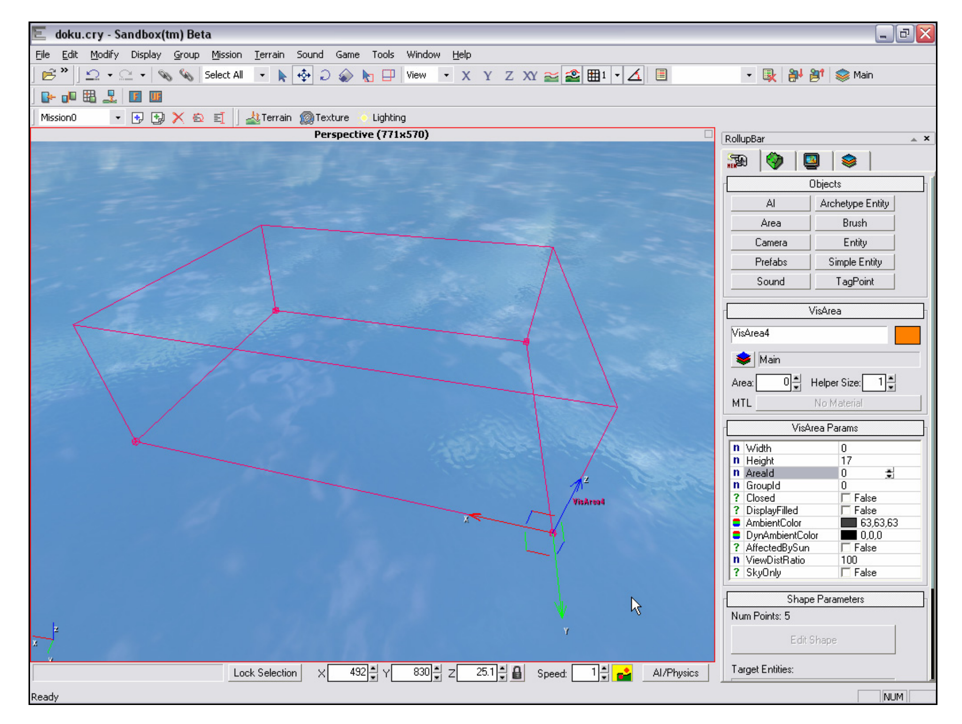


Abbildung 24: Die Höhle von Außen. Nach Aufbau einer *VisArea*

Auf Abbildung 24 wird deutlich, dass alles innerhalb einer solchen Area von Außen nicht mehr sichtbar ist.

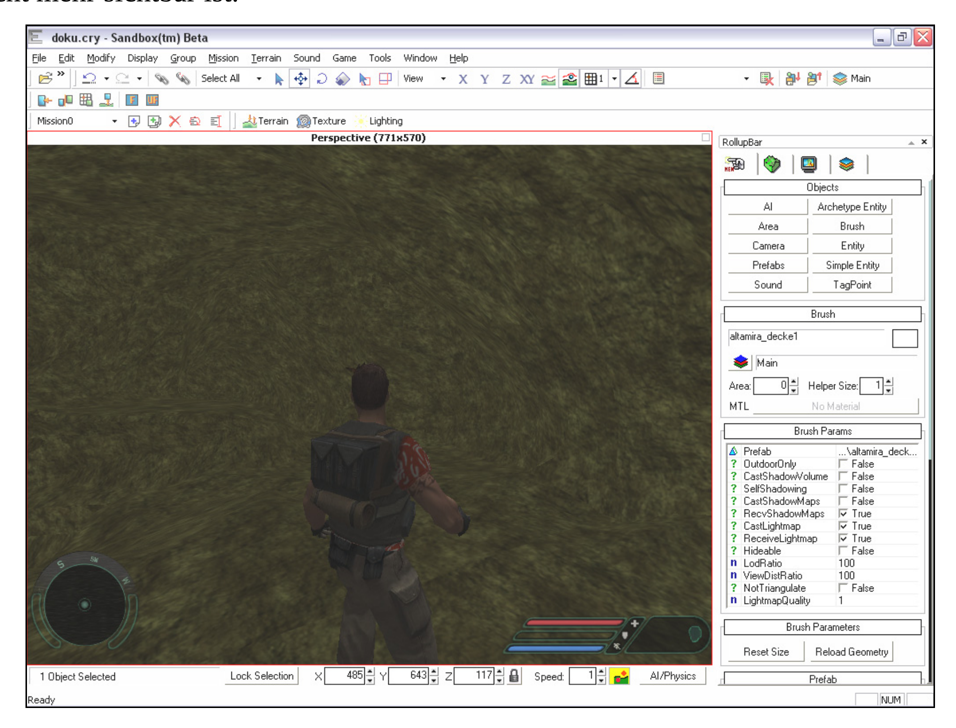


Abbildung 25: Das Innere der Höhle. Innerhalb der *VisArea*

Das Innere der Höhle ist nun nicht mehr von Außen beleuchtet. Zur Illustration ist allerdings die Abbildung etwas aufgehellt.

Nun sind alle Voraussetzungen geschaffen, um die Höhle wie gewünscht beleuchten zu können.

3.2 *Simulation verschiedener Beleuchtungssituationen*

Im Folgenden Kapitel werden einige Beleuchtungssituationen dargestellt, die sich mit dem Editor realisieren lassen. Generell gilt aber, dass dynamische Lichtquellen viel Prozessorleistung beanspruchen, so dass, je nach verwendeter Hardware, nicht zu viele dynamische Lichter eingesetzt werden sollten, da sonst der Bildaufbau stark verlangsamt wird.

3.2.1 Beleuchtungssituation 1: Mehrere kleine Lichtquellen am Boden[25]

In der einschlägigen Literatur zu prähistorischer Höhlenkunst wird oft davon ausgegangen, dass die damaligen Künstler Lampen aus Ton oder Steine mit einer Aushöhlung verwendeten. Diese Aushöhlung war gefüllt mit Knochenmark als Brennstoff, als Docht dienten Pflanzenfasern, die ebenfalls in Knochenmark getränkt waren. Mehrere dieser Lampen wurden auf dem ganzen Höhlenboden verteilt, so dass die ganze Decke gleichmäßig ausgeleuchtet war.[26]
Wie man sich das vorzustellen hat, zeigt die nun folgende Abbildung.

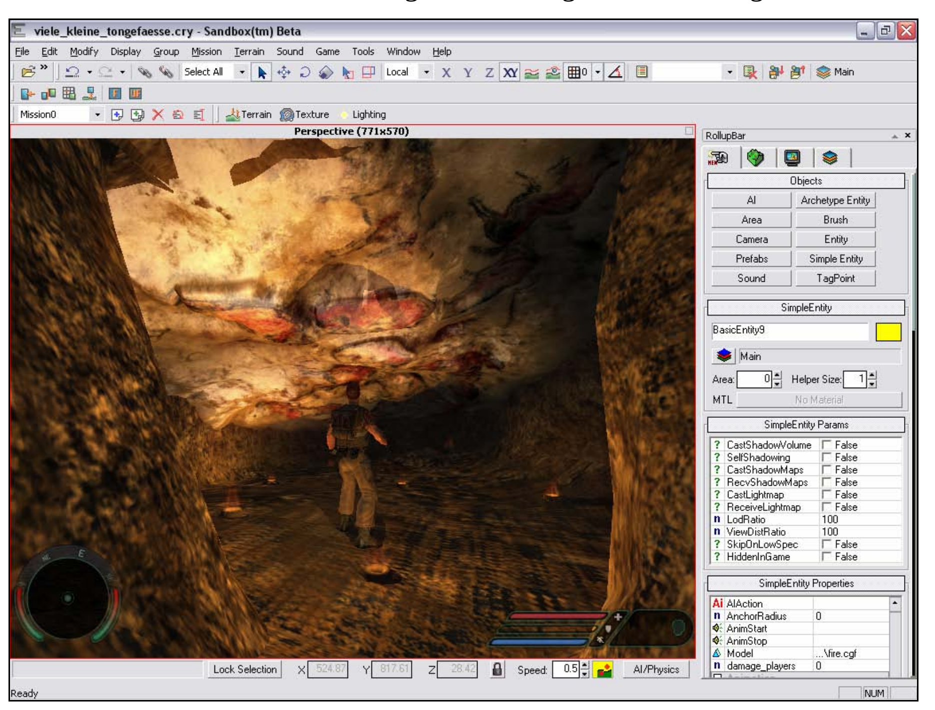


Abbildung 26: Beleuchtungssituation 1: Mehrere Tongefäße, auf dem Höhlenboden verteilt

Natürlich kann hier noch die Beleuchtungsintensität, die Reichweite der einzelnen Lichtquellen, sowie die Lichtfarbe und die Art des Lichtes (flackern, sich bewegend

---

[25] Für jede Beleuchtungssituation steht auf der beiliegenden DVD ein vorgefertigter Level zur Verfügung um die geschilderten Szenarien selbst nachvollziehen zu können.
[26] Vgl. Beltrán, Altamira, 1998. S. 67 ff.

usw.) verändert werden. Je nach eingestellter Lichtfarbe verändert sich der Gesamteindruck der Deckentextur, die oben auf Grund der eingestellten Colorierung gelblich wirkt.

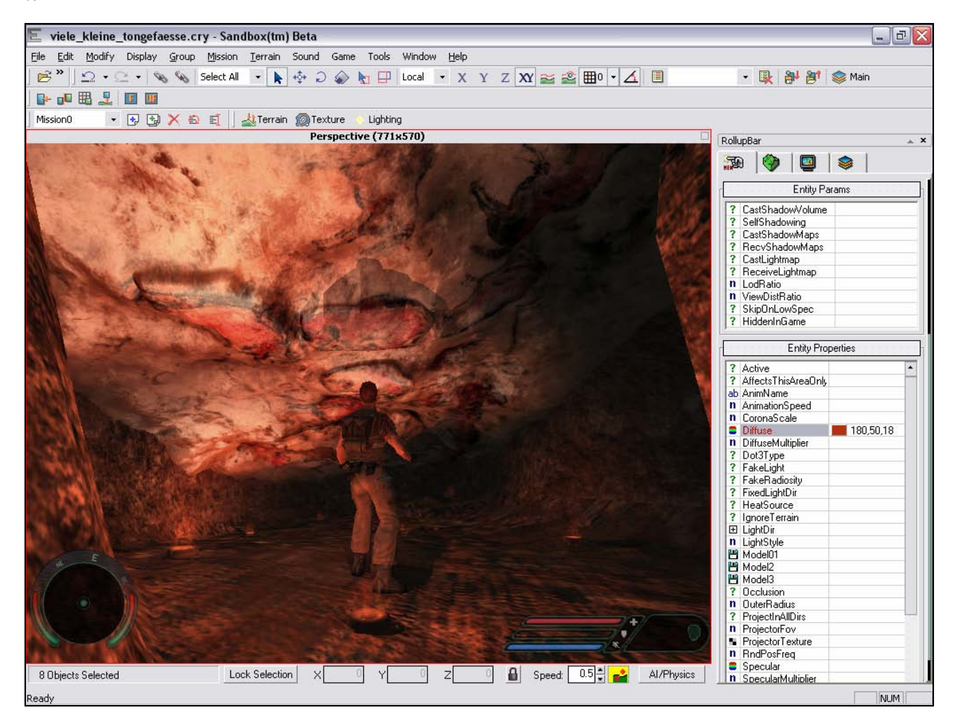


Abbildung 27: Veränderte Lichtfarbe (rot)

Außerdem lässt sich über den sog. *DiffuseMultiplier* das Ausmaß der Wirkung der Lichtfarbe auf die Textur einstellen.

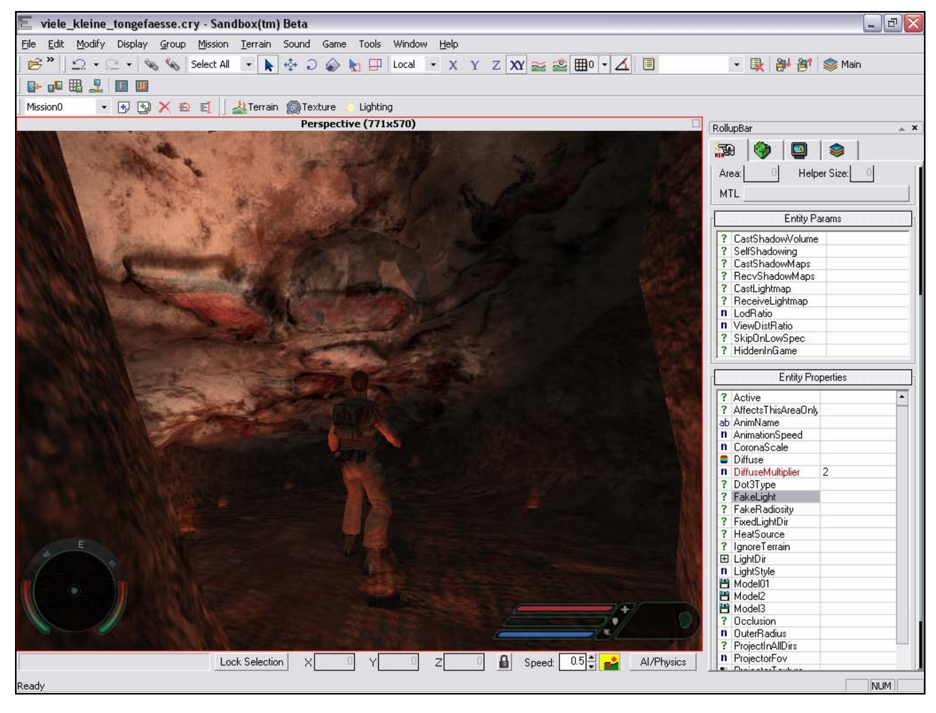


Abbildung 28: Gleiche Lichtfarbe, niedrigerer DiffuseMultiplier

Die Möglichkeiten der Manipulation der Lichtquellen sind hier nahezu unerschöpflich, da es zusätzlich die Möglichkeit gibt, sowohl vordefinierte *Lightshader* einzusetzen,

als auch selbst solche *Shader zu* schreiben, bzw. vorhandene an die eigenen Bedürfnisse anzupassen.
Aber schon in dieser relativ einfachen Beleuchtungskonstellation wird deutlich, dass es den prähistorischen Künstlern möglich gewesen sein könnte, mit mehreren kleinen Lichtquellen eine gleichmäßige Ausleuchtung der Decke zu erzielen.[27]

3.2.2 Beleuchtungssituation 2: Ein größeres Feuer in der Höhlenmitte

Die folgende Beleuchtungssituation beinhaltet eine Lichtquelle, ein Lagerfeuer in der Mitte des Raumes. Auf der Abbildung wird deutlich, dass die Ausleuchtung der Höhle wesentlich unregelmäßiger erfolgt, als in der vorherigen Situation.

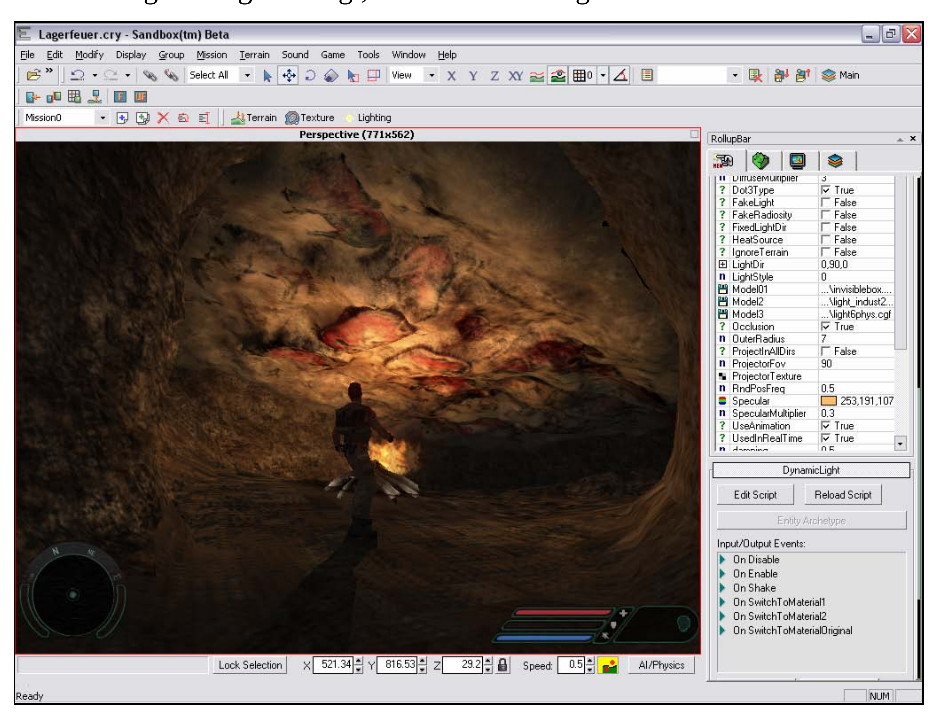


Abbildung 29: Beleuchtung der Höhle durch ein Lagerfeuer

Um der Szene etwas mehr Realismus einzuhauchen, wurde zusätzlich ein Partikeleffekt als Lagerfeuer eingesetzt. Dieser dient aber nicht der Beleuchtung, sondern ist nur grafische Ausschmückung. Auch hier bieten sich viele Möglichkeiten, die Beleuchtung zu verändern. Mit wenigen Handgriffen kann man das Licht flackern oder auch sich leicht hin und her bewegen lassen, was zu sich bewegenden Schatten führen würde.

---

[27] Als kleine Einschränkung sollte darauf hingewiesen werden, dass ab einer gewissen Anzahl dynamischer Lichtquellen, scheinbar aufgrund eines Fehlers im Editor, einzelne Lichtquellen nicht richtig dargestellt werden, bzw. erst dann dargestellt werden, wenn man sich mit der Spielfigur bis auf eine bestimmte Entfernung genähert hat (siehe die beiden „hinteren Lichter“ in dieser Beleuchtungssituation).

Unter Verwendung nur einer Lichtquelle erscheint es plausibler, dass die Anordnung der Malereien auf die Beleuchtung zurück zu führen ist, da Teile der Höhlendecke immer im Schatten liegen. Zusätzlich wirft auch der Künstler selbst einen Schatten auf die Decke, da sich die Hauptlichtquelle am Boden befindet.

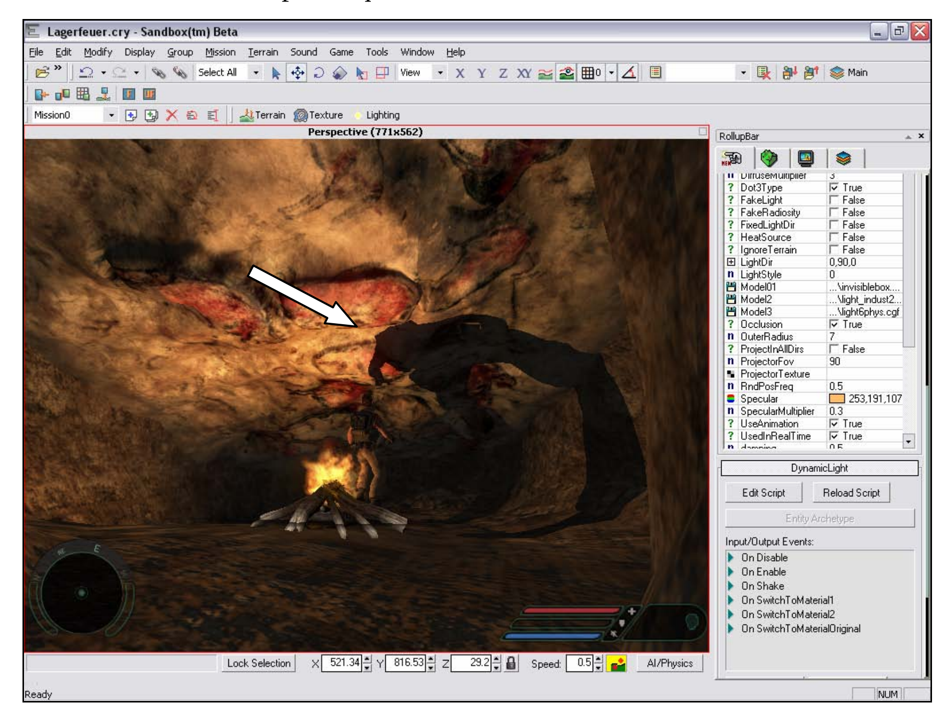


Abbildung 30: Schattenwurf der Spielfigur verdeckt teilweise die Malereien an der Decke

3.2.3 Beleuchtungssituation 3: Versuch einer beweglichen Lichtquelle

Um die Ausgangshypothese dieser Arbeit im Detail zu prüfen, wäre es am sinnvollsten, eine bewegliche Lichtquelle zu verwenden, z.B. eine Fackel oder ähnliches, das die Spielfigur umher tragen kann. So könnte man nachvollziehen, was ein prähistorischer Künstler gesehen haben mag, wenn er mit einer Fackel in der Hand unter der polychromen Decke stand.

Leider stößt die *CryEngine* hier vorerst an ihre Grenzen. Zumindest scheint es, mit den Standardmitteln des Editors, nicht ohne weiteres möglich zu sein. Zwar gibt es eine vordefinierte tragbare Lichtquelle, eine Taschenlampe, aber leider erzeugt diese keine wirklich dynamische Beleuchtung. Vielmehr projiziert sie eine Lichttextur, die dort, wo sie hinfällt, die darunter liegenden Pixel der Textur aufhellt. Außerdem wirft die Taschenlampe keine Schatten.

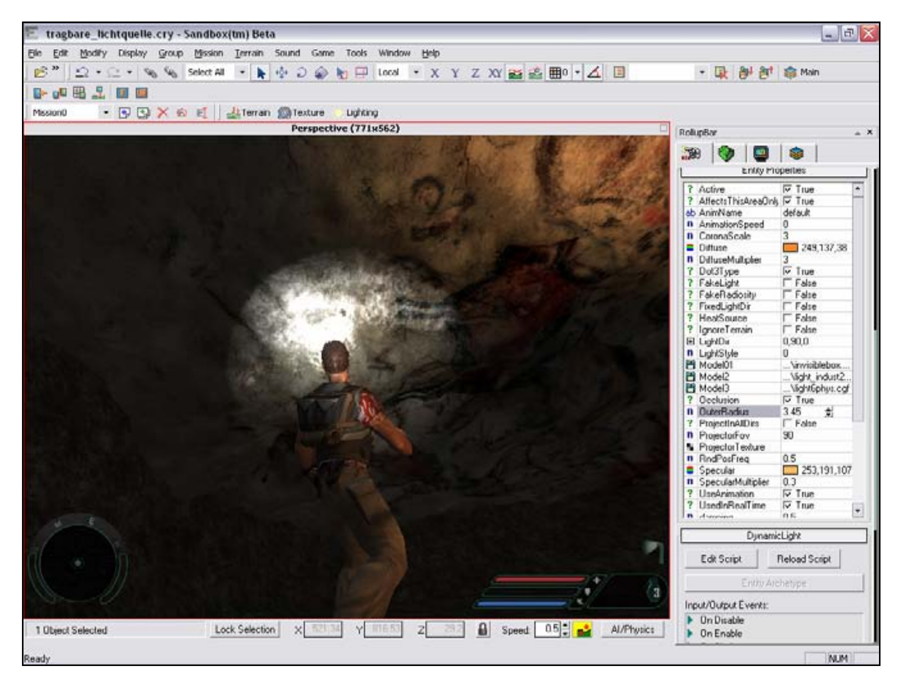


Abbildung 31: Lichtkegel der Taschenlampe

Im oberen Bild sieht man den Lichtkreis, den die Taschenlampe[28] erzeugt. Leider kann hier auch nicht die Lichtfarbe und die Lichtintensität angepasst werden.

Als wirklich realistisch einsetzbare Lichtquelle scheidet die Taschenlampe also aus.

Aber es gibt eine Alternative. Diese besteht in der kombinierten Verwendung der Editor- und der In-Game-Ansicht. So wäre es vorstellbar, im Editor eine kleine und schwache Lichtquelle unterhalb der Decke zu positionieren, etwa dort, wo sie sich befände, wenn sie von einem Menschen gehalten würde.

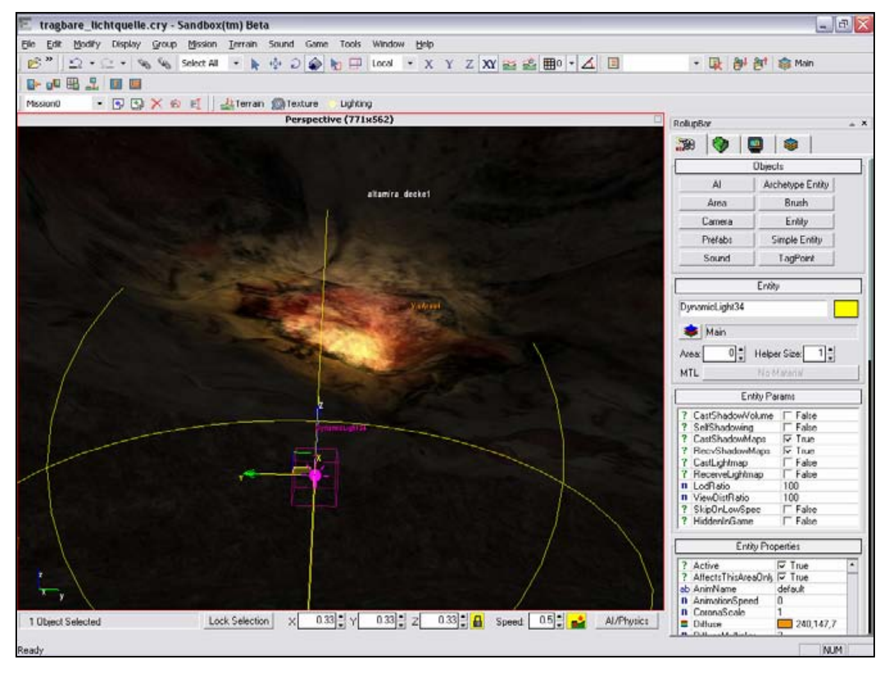


Abbildung 32: Ansicht einer Lichtquelle im Editor

[28] Im Editor wird die Taschenlampe im In-Game-Modus (Strg-G) mit der Taste „L“ eingeschaltet.

Per Tastendruck (Strg-G) kann man jederzeit aus der Editor in die Ingame-Ansicht umschalten um das Ergebnis zu überprüfen. Eine solche Lichtquelle kann, wie in den vorangegangenen Beispielen schon beschrieben, in vieler Hinsicht manipuliert werden. Zwar ist diese Lösung nicht ganz so elegant wie gewünscht, bietet aber dennoch viele Möglichkeiten, um auch tragbare Lichtquellen zu simulieren, vor allem, da man bereits im Editor die Wirkung der Lichtquelle überprüfen kann.

3.2.4 Beleuchtungssituation 4: Experimentelle Beleuchtung mit halbbeweglicher Lichtquelle

Als letzte Variante soll nun noch eine eher experimentelle Beleuchtungssituation dargestellt werden.

Auch wenn es bisher nicht geglückt ist, eine voll bewegliche und tragbare Lichtquelle zu schaffen, so ist es dennoch möglich, wenn auch bisher nur in einem frühen experimentellen Stadium, zumindest halb bewegliche Lichtquellen zu integrieren. Dazu ist ein Plugin aus der Entwickler Community erforderlich. Dieses Plugin nennt sich *PickMeUp*[29]. Es ermöglicht normalerweise, sog. *physikalisierte Objekte* zu bewegen, bzw. von der Spielfigur tragen zu lassen. Ein physikalisiertes Objekt wäre z.B. ein Fass, das vom Spieler umgeworfen werden kann und welches dann, physikalisch korrekt, einen Berghang hinunterrollt. Ermöglicht wird dies durch die in die CryEngine integrierte Physikengine.

Der Editor bietet mehrere vorgefertigte Lichtquellen an, unter anderem z.B. auch eine Deckenleuchte, die an einem Kabel von der Decke hängt. Diese Deckenleuchte kann von der Spielfigur angestoßen werden und schwingt dann hin und her, wobei sie die Umgebung entsprechend dynamisch beleuchtet.

In diesem experimentellen Aufbau soll nun eine solche Deckenlampe verwendet werden, um die Möglichkeit zu schaffen, Lichtquellen direkt im Spiel, wenn auch in eingeschränktem Rahmen, beweglich zu machen.

Zugegebenermaßen ist eine Deckenleuchte natürlich alles andere als „zeitgenössisch“, dennoch soll sie hier dazu dienen, weitere Möglichkeiten der Beleuchtung zu veranschaulichen.

---

[29] Die sog. PickMeUp-Entity von Aarbro. Dieses Add-On kann heruntergeladen werden unter folgender Adresse:
Quelle: Die offizielle Crytek-Community Webseite, 09.02.05:
http://www.crymod.com/index.php?templateid=showposts&fid=10&tid=1497
oder per Direkt-Link: http://web.telia.com/~u31225218/fc/PickMeUp_TestVersion.zip. Der Foren-Eintrag liegt dieser Arbeit im pdf-Format bei (auch das Plugin selbst liegt auf der DVD bei.).

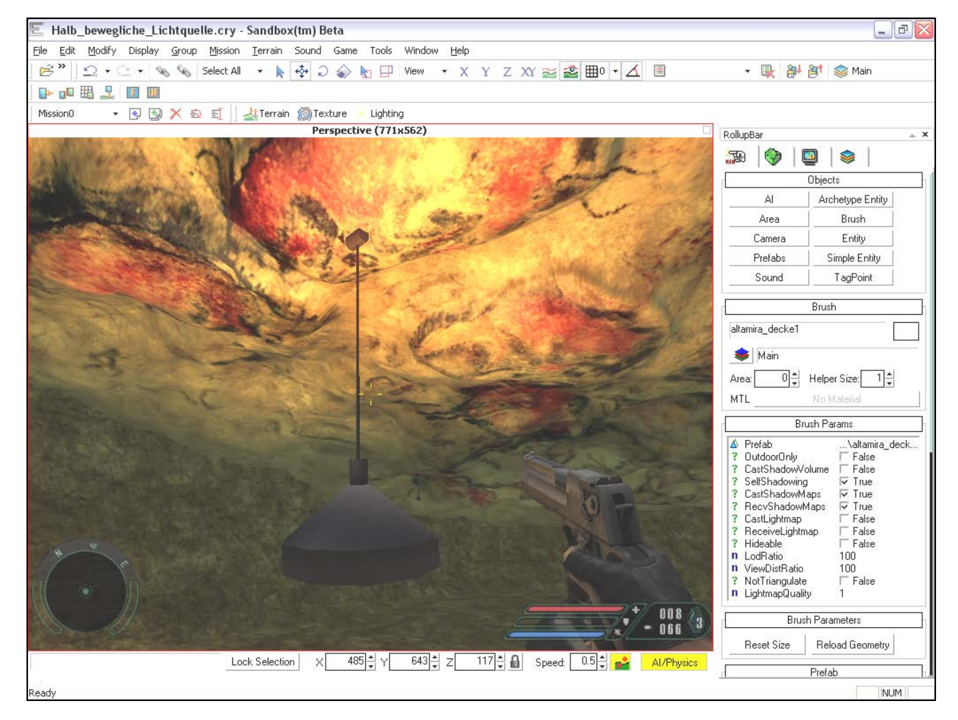


Abbildung 33: Bewegliche Deckenlampe

Es gibt mehrere Wege, die Deckenlampe in Bewegung zu versetzen. Der einfachste ist, sie durch einfaches Berühren schwingen zu lassen.
Als zweite Möglichkeit steht die *PickMeUp-Entity* zur Verfügung. Hier kann man im In-Game-Modus durch einmaliges Drücken der mittleren Maustaste das Objekt greifen und festhalten. Bewegt man nun die Spielfigur, zieht sie die Lampe hinter sich her. Durch Drücken der Taste „t“ kann das Objekt weggeworfen werden, mit der Taste „r“ kann man kurz das Bild anhalten.
Die dritte Möglichkeit mag etwas martialisch anmuten, erzeugt aber ebenfalls recht ansehnliche Beleuchtungseffekte: In diesem Level steht der Spielfigur eine Waffe zur Verfügung, mit der man auf die Deckenleuchte schießen kann, um sie so in Bewegung zu versetzen.
Diese letzte Beleuchtungsvariante zeigt, dass die *CryEngine* noch ausbaufähig ist. Bestimmt ist es nur eine Frage der Zeit, bis sich auch tragbare Lichtquellen realisieren lassen. Aus der großen Benutzer-Gemeinschaft kommen ständig neue Vorschläge und Ideen zur Erweiterungen der Game-Engine. Die *PickMeUp-Entity* ist bereits in Ihrer frühen Version ein viel versprechender Ansatz.

4 Fazit

Die Frage, ob die verwendeten Software-Komponenten sich in einem wissenschaftlichen Umfeld zur Rekonstruktion und Analyse verlorener Räume eignen, wird im Folgenden unter zwei Gesichtspunkten betrachtet: Wie groß ist die benötigte Be- und Einarbeitungszeit? Und in wie weit eignet sich die verwendete Game-Engine zur Simulation und Analyse verlorener Räume?

4.1 Be- und Einarbeitungsaufwand

Um beurteilen zu können, ob die hier gewählte Kombination von Softwarekomponenten in einem wissenschaftlichen Umfeld sinnvoll eingesetzt werden kann, ist vor allem auch die Frage zu klären, wie viel Be- und Einarbeitungszeit ein solches Projekt erfordert und ob der betriebene Aufwand im Verhältnis zum erzielten Ergebnis steht.

3ds Max ist eine sehr umfangreiche und komplexe Applikation, die einige Zeit der Einarbeitung erfordert. Hier steht aber eine Fülle von hilfreicher Literatur zur Verfügung, die es auch Anfängern schnell ermöglicht, erste Ergebnisse zu erzielen. Zudem sind, zumindest für ein Projekt wie dieses, in erster Linie Grundkenntnisse in *3ds Max* erforderlich, die mit Hilfe unterstützender Literatur rasch erlernbar sind. Auch das Internet bietet eine reiche Auswahl an Foren zu *3ds Max* an, in denen man Fragen an erfahrene Benutzer richten kann, die einem Neuling in den meisten Fällen schnell und unkompliziert weiterhelfen. Begünstigt wird dies durch die weite Verbreitung von *3ds Max*.

Was die zur Verfügung stehende Literatur betrifft, ist der *Sandbox-Editor* momentan leider noch etwas unterversorgt. Da es sich um ein Produkt handelt, das sich nicht in erster Linie an wissenschaftliche Kreise richtet, liegt bislang auch noch keine Sekundärliteratur in Buchform vor. Dafür finden sich im Internet zahlreiche *Tutorials* zu diesem Thema, und auch hier gibt es eine Reihe von Internetseiten und Foren mit einer wachsenden Community, die bei Fragen und Problemen meist schnelle Hilfe leistet.

Da der Editor darüber hinaus von der Bedienoberfläche stark an *3ds Max* angelehnt ist, findet man sich hier schnell zurecht, hat man sich erstmal in *3ds Max* eingearbeitet.

Was also den Bearbeitungsaufwand angeht, so ist es durchaus möglich, sich innerhalb von 3 bis 4 Monaten in beide Softwarepakete einzuarbeiten und sie sinnvoll zu verwenden.

Für ein späteres Projekt in einem institutionellen oder universitären Umfeld bliebe, je nach Budget, auch die Möglichkeit, die Modellierungsarbeiten an einen professionellen 3D-Designer abzugeben, was zum einen die Erstellung des Modells wesentlich beschleunigen würde und zum anderen die Anschaffung der relativ kostspieligen Software überflüssig macht, sofern sie nicht schon vorhanden ist.

Für die Verwendung einer fertigen Game-Engine, die zusätzlich noch einen komfortablen und mächtigen Editor zur Verfügung stellt, spricht vor allem der Zeitfaktor, da man nicht selbst eine 3D-Engine programmieren muss. Ein solches Vorhaben beansprucht in jedem Fall wesentlich mehr Zeit, als die Einarbeitung in ein fertiges und getestetes Produkt.[30]

4.2 Verwendbarkeit der Game-Engine zur Analyse verlorener Räume

Eingangs wurde die Hypothese aufgestellt, dass die palimpsestartige Anordnung der Kunstwerke an der polychromen Decke aus der damaligen Beleuchtungssituation resultieren könnte.[31] An dieser Hypothese sollte die Verwendbarkeit der gewählten Game-Engine für wissenschaftliche Zwecke überprüft werden.

Anhand der oben dargestellten Beleuchtungssituationen sind bereits viele der Möglichkeiten deutlich geworden, die die gewählte Engine bietet. Speziell für das Thema dieser Arbeit liefert die gewählte Software-Kombination ansehnliche und rasche Resultate, ohne eigene Programmierarbeit zu erfordern.

Die dynamischen Lichtquellen lassen sich in vielerlei Hinsicht den Bedürfnissen des Forschers oder des Benutzers anpassen, um eine realistische Beleuchtung der Höhle zu ermöglichen.

Über die in den Editor integrierte Scriptssprache LUA können sowohl eigene, wie auch mitgelieferte Objekte manipuliert werden, ohne Einfluss auf den Sourcecode zu nehmen, wodurch die Engine sehr flexibel erweiterbar ist. Das vorgestellte *PickMeUp-Add-On* ist hierfür nur ein Beispiel. Mit eingehender Einarbeitung in diese Scriptsprache sollte es auch möglich sein, eine voll bewegliche, dynamische Lichtquelle zu implementieren. Leider reichte hierfür die zur Verfügung stehende Zeit nicht aus.

Trotzdem hat sich gezeigt, dass die verwendete Software hinsichtlich dieser Hypothese durchaus eine wissenschaftliche Analyse des Raums ermöglicht.

Auch bietet die integrierte Physikengine weitere Möglichkeiten, die im Rahmen dieser Arbeit nicht benötigt wurden. Vorstellbar wäre für die Zukunft eine noch detailliertere

---

[30] Rechtlich sollte wahrscheinlich nichts dagegen sprechen, den Editor für wissenschaftliche Zwecke zu nutzen, leider erfolgte jedoch von *Crytek* bislang noch keine konkrete Antwort auf diese Frage.

[31] Es soll an dieser Stelle noch einmal darauf hingewiesen werden, dass sowohl die verwendete Höhle, als auch die Fragestellung hinsichtlich der Beleuchtung, für diese Arbeit nur exemplarischen Charakter haben. So ist sich die Literatur zu prähistorischer Höhlenmalerei einig, dass die Anordnung der sich stellenweise überlagernden Kunstwerke *nicht* auf die Beleuchtungssituation zurückzuführen ist.

Vgl. hierzu: Lorblanchet, Höhlenmalerei, 1997. S. 289-293. Vielmehr werden hier zwei Interpretationsansätze geliefert. Zum einen wird davon ausgegangen, dass die Anordnung, sowie eventuelle Überschneidungen der Malereien und Zeichnungen ein Resultat zeitlicher Abfolge sind. Der zweite Ansatz ist eher strukturalistisch zu verstehen. Er geht davon aus, dass die Überschneidungen mit Absicht von dem oder den Künstler/n gesetzt sind, um eine Zusammengehörigkeit der Motive zu betonen.

Nachbildung der Höhle, mit hochauflösenderen Texturen (soweit verfügbar) für die Decke und einer feiner ausgestalteten Felstextur für die restliche Höhle. Dieses Modell könnte im Sandbox-Editor in eine virtuelle Landschaft eingesetzt werden, so dass der Benutzer am PC die Umgebung der Höhle und schließlich die Höhle selber erkunden kann und zwar in einem Zustand, der dem Entstehungszeitpunkt der Malereien entspricht. Interessant wäre dies zum Beispiel für ein Museum, wo Besucher dieses Höhlenmodell an einem Computerterminal erforschen könnten.[32]

Die Möglichkeiten sind hier kaum eingeschränkt. Einzig die Limitierung der Engine auf ca. 100.000 Polygone pro Szene setzt der Kreativität gewisse Grenzen. Allerdings hat das hier verwendete Höhlenmodell insgesamt nur ca. 15.000 Polygone und macht dennoch einen respektablen Eindruck. Es wäre also durchaus möglich, auch die komplette Höhle nachzubauen. Sofern detailliertere Geometriedaten zur Verfügung stehen, könnte das Modell auch noch weiter an die Realität angepasst werden.

Aber die Engine eignet sich nicht nur für solche Zwecke. Einer ihrer größten - und hier bisher nicht genutzten - Vorteile besteht in der realistischen Darstellung großer Außen-Areale mit hoher Sichtweite. Man könnte also auch prähistorische Siedlungsplätze rekonstruieren oder Antike Städte „wiederaufbauen“, wie z.B. Pompeji vor der Zerstörung, um sie interaktiv begehbar zu machen.

Abschließend lässt sich sagen, dass die fortschreitende Entwicklung auf dem Gebiet der 3D-Grafik, begünstigt durch immer leistungsfähigere Hardware, durchaus sinnvoll für die Präsentation und Analyse wissenschaftlicher Forschungsergebnisse genutzt werden kann.

[32] Allerdings wären hierfür wahrscheinlich einige lizenzrechtliche Punkte zu klären.

5 Installationsanleitung

Das folgende Kapitel liefert eine eingehende Beschreibung der Installation der verwendeten Software. Alle benötigten Komponenten sowie zusätzliches Informationsmaterial liegen dieser Arbeit auf DVD bei.

5.1 Hardwareanforderungen

Um den Sandbox-Editor sinnvoll zu nutzen und die beigelegten Level in der intendierten Qualität betrachten zu können wird folgende Hardware benötigt:

> Empfohlene Systemanforderungen
> - Computer: 2GHz AMD-Prozessor oder höher
> - Speicher: 512 MB RAM [oder mehr]
> - Laufwerk: 4 GB freier Speicherplatz auf Ihrer Festplatte für Installation und Ausführung des Programms. 8fach DVD oder 32fach CD-ROM.
> - Grafik: 3D-Beschleunigerkarte (Ab Nvidia GeForce4 TI 4600 oder vergleichbar, die DirectX® 9.0b unterstützt, mit mindestens 128 MB Grafikspeicher.
>
> Hohe Systemanforderungen
> - Computer: 3 GHz AMD-Prozessor oder höher
> - Speicher: 1024 MB RAM
> - Laufwerk: 4 GB freier Speicherplatz auf Ihrer Festplatte für Installation und Ausführung des Programms. 16fach DVD oder 32fach CD-ROM.
> - Grafik: 3D-Beschleunigerkarte (Ab ATI Radeon 9800Pro oder vergleichbar), die DirectX® 9.0b unterstützt, mit mindestens 128 MB Grafikspeicher.[33]

Bei der Erstellung der vorliegenden Arbeit wurden folgende Hardware und Treiber verwendet:

- Computer: AMD Athlon XP 2800+
- 1024 MB RAM
- Grafik: Nvidia GeForce4TI 4600, mit 128 MB Speicher
- Grafiktreiber: Nvidia ForceWare 66.93
- Microsoft DirectX9c (liegt auf DVD bei)

[33] Diese Angaben sind der offiziellen, dem Spiel beiliegenden ReadMe-Datei entnommen.

5.2 FarCry

Da der Editor und das SDK auf die Ressourcen des Spiels zurückgreifen, muss dieses zuerst installiert werden.

Legen Sie dazu bitte die mitgelieferte FarCry-DVD in ein DVD-Laufwerk und starten Sie den Installationsvorgang (sofern Sie nicht die Autostartfunktion des Laufwerks eingeschaltet haben) durch Doppelklick auf die ***setup.exe*** im Quellverzeichnis der DVD.

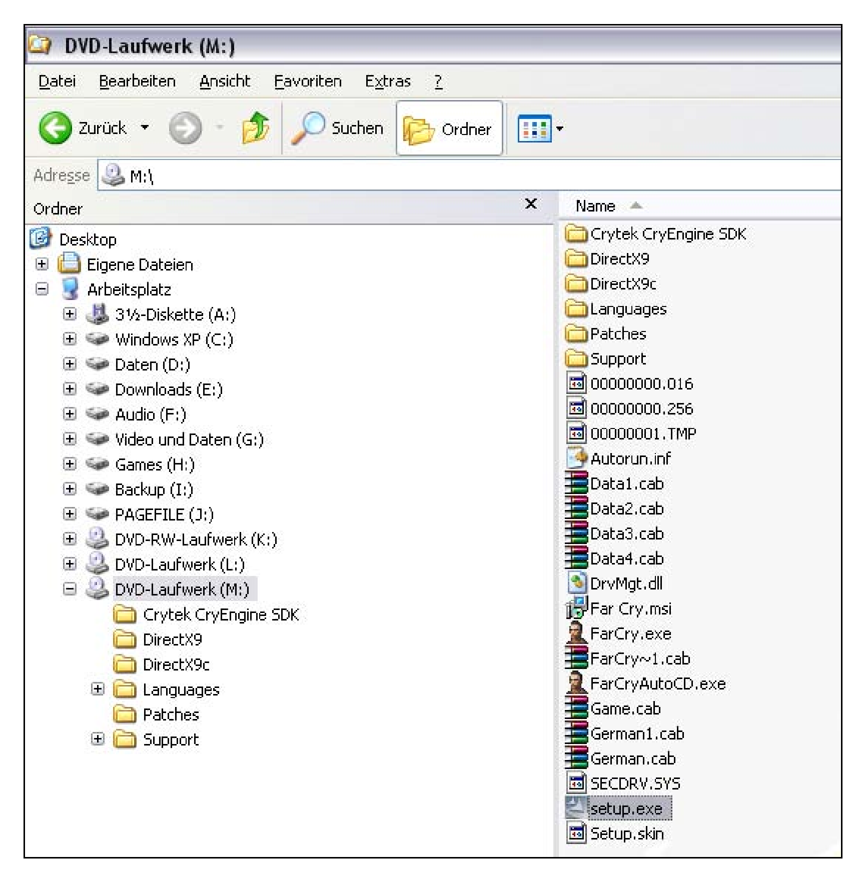


Abbildung 34: Starten der setup.exe durch Doppelklick

Folgen Sie den Anweisungen des Installationsprogramms und wählen Sie im Verlauf der Installation das Ziellaufwerk und –verzeichnis aus.

5.3 FarCry-Patch v1.3

Nach erfolgreicher Installation von ***FarCry*** sollte der beilgelegte Patch v 1.3 installiert werden, um das Spiel und den Editor auf den neuesten Stand zu bringen.

Der Patch befindet sich in folgendem Verzeichnis:

DVD-Laufwerk:\Patches

Starten Sie den Installationsvorgang durch Doppelklick auf die folgende Datei:

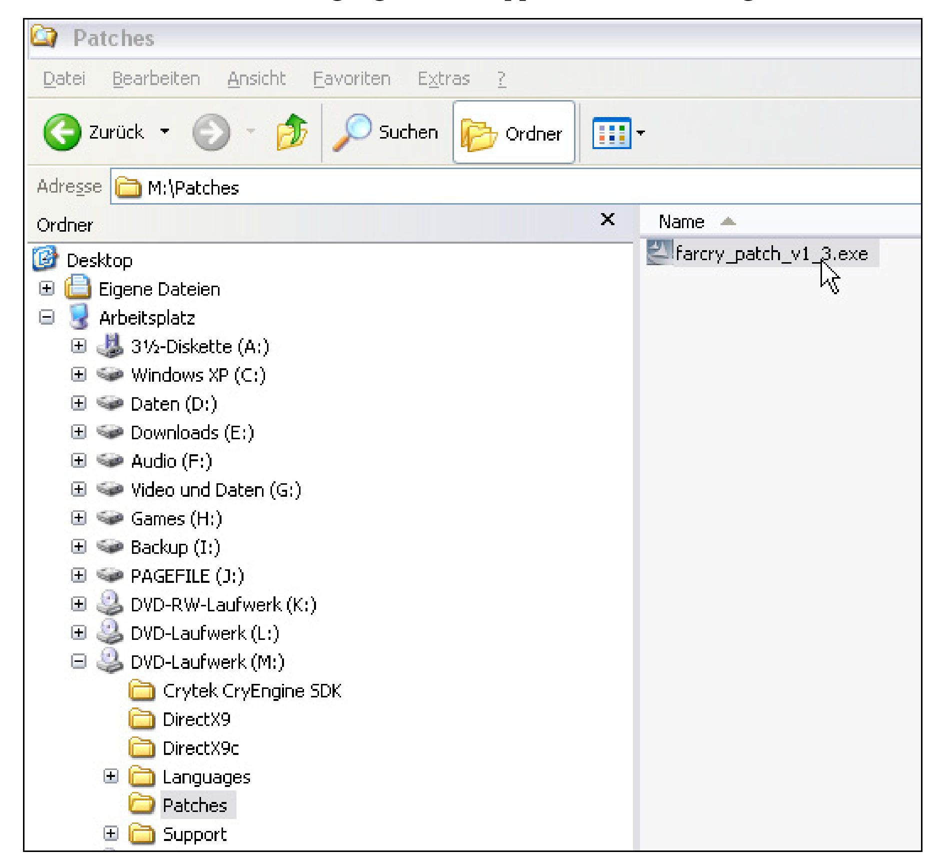


Abbildung 35: Installation des FarCry-Patch v 1.3

Das Installationsprogramm sollte selbstständig das Verzeichnis auswählen, in welches das Spiel zuvor installiert wurde.

Befolgen Sie die Anweisungen der Setup-Progamms.

5.4 Microsoft DirectX9c

Ist die Installation beendet sollten Sie, sofern noch nicht auf ihrem System vorhanden, ***Microsoft DirectX9c*** installieren. Dieses liegt auf der FarCry-DVD in folgendem Verzeichnis bei:

DVD-Laufwerk:\DirectX9c\.

Starten Sie die Installation durch Doppelklick auf die ***directx_9c_redist.exe*** und folgen Sie den Anweisungen des Installationsprogramms.

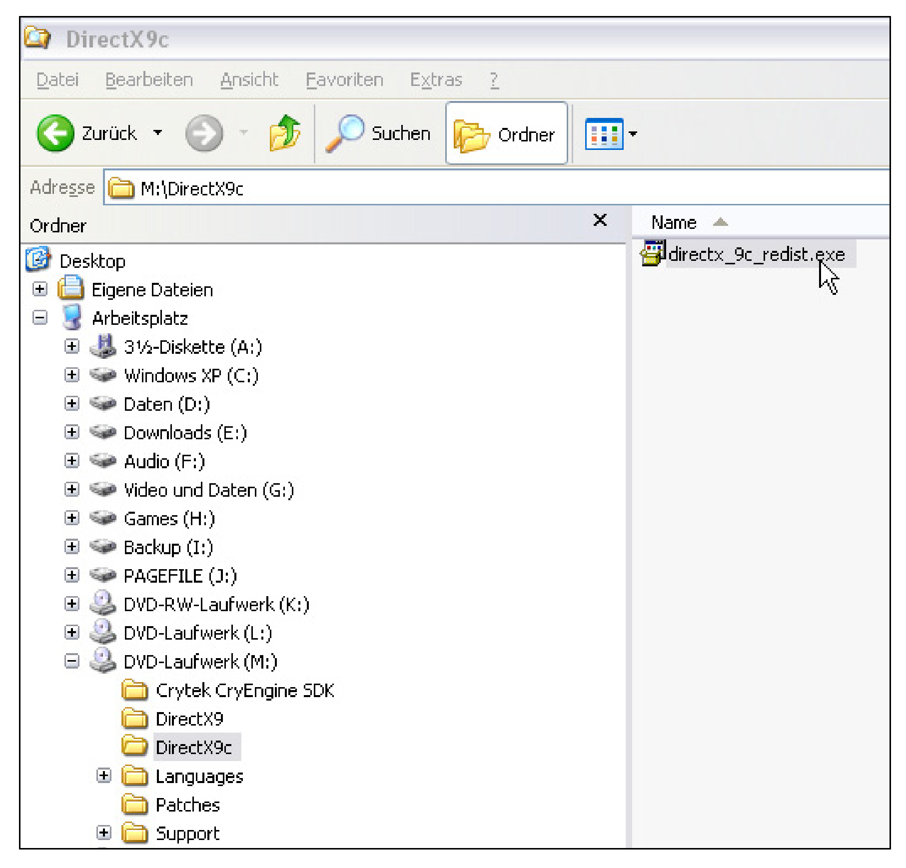


Abbildung 36: Installation von Microsoft DirectX9c

Eventuell muss der Computer nach der Installation von DirectX9c neu gestartet werden.

5.5 Crytek CryEngine SDK v1.1

Um eigene Modelle in den Sandbox-Editor zu importieren, benötigt man das CryEngine SDK. Dieses liegt in der neuesten Version 1.1 dieser Arbeit auf der FarCry-DVD in folgendem Verzeichnis bei:

DVD-Laufwerk:\Crytek CryEngine SDK \CryENGINE_MOD_SDK_11_Setup.exe.

Im Folgenden eine kurze Beschreibung des MOD SDK:

> Das CryEngine™ MOD SDK beinhaltet eine Auswahl von Tools bestehend aus Exportern, Plug-Ins und Quell-Code die dem Benutzer erlauben das Far Cry™ Spielerlebnis durch den Import von neuen Inhalten wie Models und Animationen zu erweitern. Der 30 MB große Download enthält eine dokumentierte Anleitung und gibt Benutzern erstmals die Möglichkeit „Total Conversion Mods" mit der CryENGINE™ Technologie zu erschaffen. Das SDK wird in Verbindung mit dem Sandbox Editor die Möglichkeit eröffnen eine schier unendliche Anzahl von Szenarien und Umgebungen zu erstellen.
>
> **Änderungen der Version 1.1:**
> - fixed Maya5 Polybump plugin
> - fixed not working antialising in Maya5 Polybump plugin
> - included Maya6 Polybump plugin
> - updated Maya Polybump .pdf file
> - included Max6 Polybump plugin
> - included ScriptObjectSystem.pdf
> - included ScriptObjectGame.pdf
> - included Max6 CryExporter
> - included Maya6 CryExporter
> - AI description document
> - Updated physics document[34]

Starten Sie die Installation des SDK durch Doppelklick auf die ***CryENGINE_MOD_SDK_11_Setup.exe*** und folgen Sie den Anweisungen des Installationsprogramms.

[34] Vgl. Ubisoft Webseite. 11.02.2005. http://www.ubi.com/DE/Downloads/Info.aspx?dlId=594.

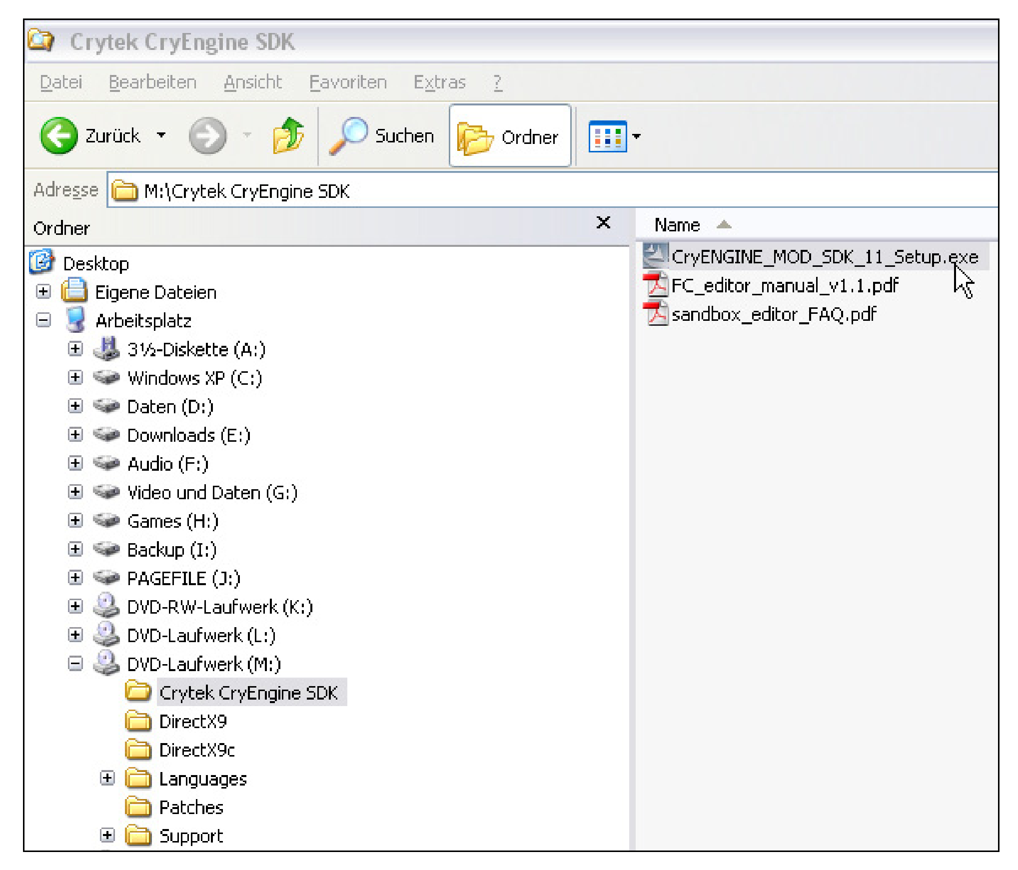


Abbildung 37: Installation des CryEngine MOD SDK v1.1

Im gleichen Verzeichnis finden Sie auch das Benutzerhandbuch des Sandbox-Editors in der Version 1.1, im .pdf-Format.

5.6 Öffnen der mitgelieferten 3ds Max Dateien mit 3ds Max

Beim ersten Öffnen der mitgelieferten .max-Dateien muss eventuell eine Pfadangabe korrigiert werden.

In einem Dialogfenster meldet 3ds Max, dass die eingebundenen Texturen nicht gefunden werden konnten.

Mit einem Klick auf den „Browse“-Button sollte man den folgenden Pfad auswählen und mit einem Klick auf den „Add“-Button der Liste hinzufügen:

[DVD-Laufw.]:\Altamira\3ds Max 6 Dateien\Texturen\

Die Texturen sollten nun richtig dargestellt werden.

5.7 Crytek Exporter Plugin für 3ds Max

Das verwendete Exporter-Plugin ist für 3ds Max 6 vorgesehen. Leider bestand keine Möglichkeit es mit 3ds Max 7, der neuesten Version zu testen. Der im Folgenden beschriebene Installationsvorgang bezieht sich also auf Version 6 von 3ds Max.
Das Plugin liegt dieser Arbeit auf der FarCry-DVD bei:

M:\Crytek CryEngine SDK\Crytek Exporter Plugin für 3ds Max 6\Max6

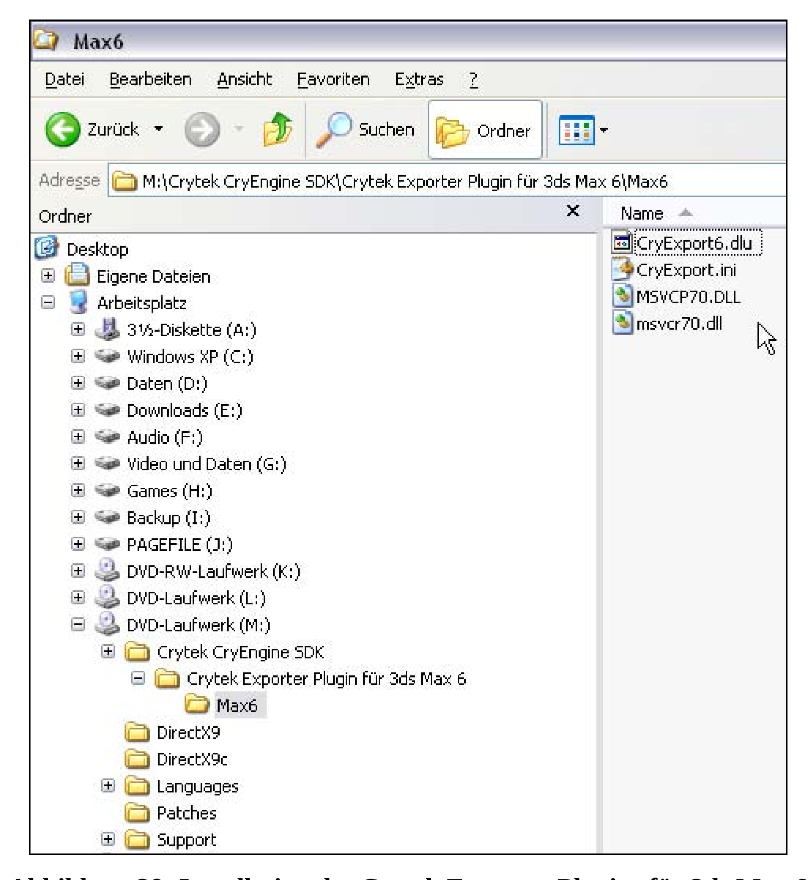


Abbildung 38: Installation des Crytek Exporter Plugins für 3ds Max 6

Zur Installation müssen die vier oben gezeigten Dateien in das ***plugins***-Verzeichnis von 3ds Max 6 kopiert werden.

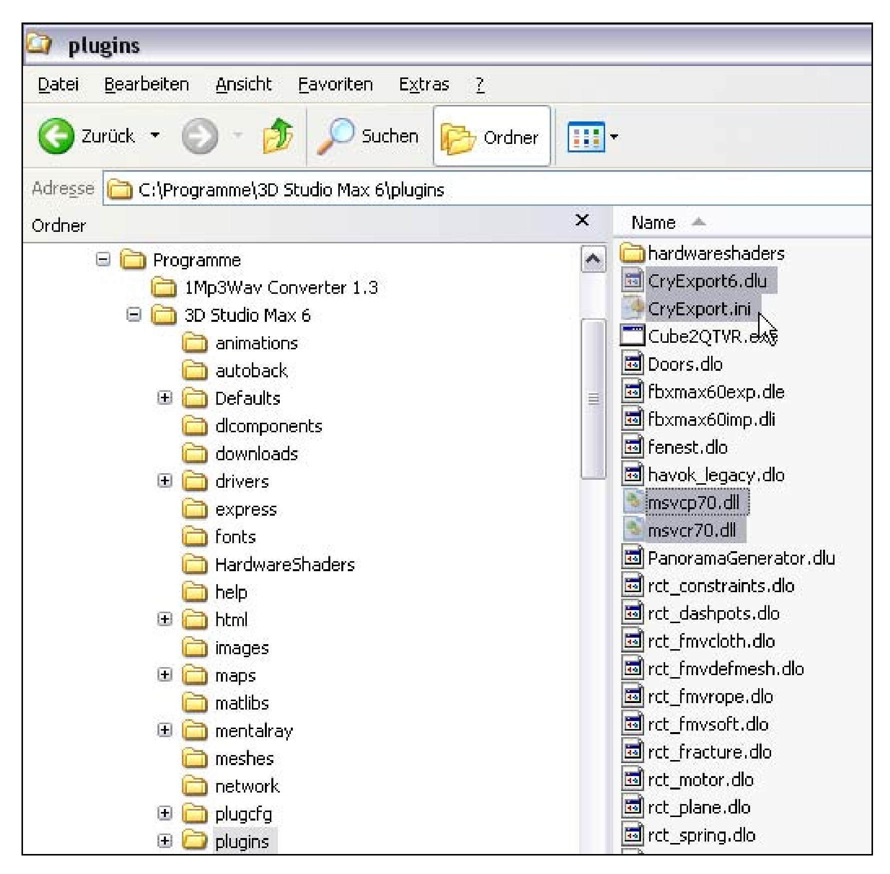


Abbildung 39: Das plugins-Verzeichnis von 3ds Max

Um das Plugin in 3ds Max zu verwenden, starten sie 3ds Max und klicken Sie, im Rollout rechts im Bild, auf das *Utilities*-Symbol (der Hammer ganz rechts).

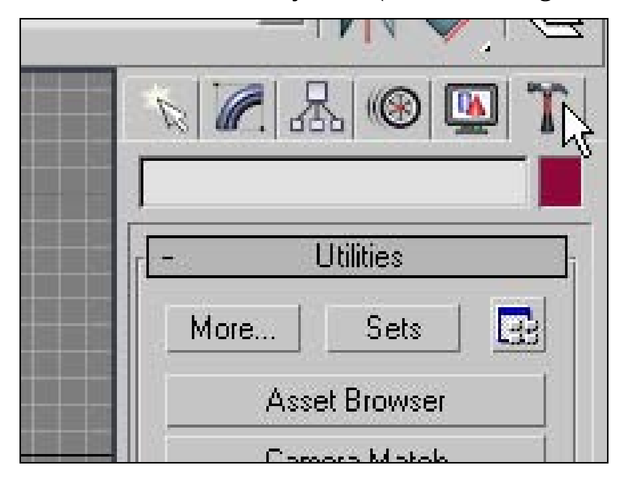


Abbildung 40: Der Utilities-Button in Max

Anschließend klicken Sie auf *More…*(im oberen Bild links) und in dem sich daraufhin öffnenden Fenster auf *Crytek Exporter2*.

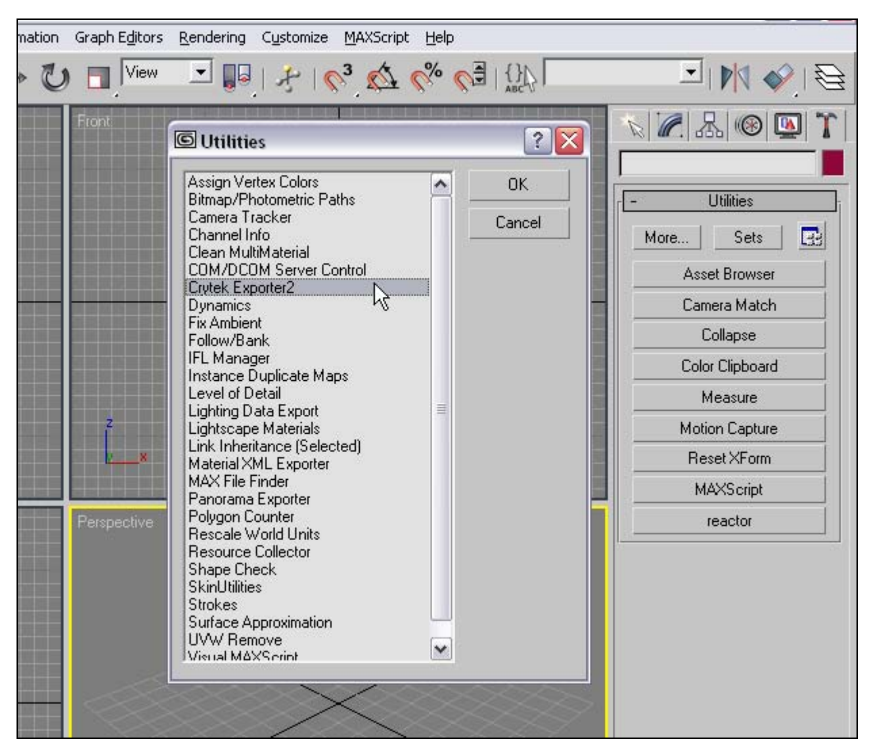


Abbildung 41: Das installierte Crytek Exporter Plugin

Mit einem Klick auf *OK* wählen Sie das Plugin zur Verwendung aus.
Für nähere Informationen liegen dem Plugin auf der DVD einige Dokumentationen im pdf-Format bei.

5.8 Das DCL-Plugin

Das DCL-Plugin (*Dynamic Class Loader*) wird in der experimentellen *Beleuchtungssituation 4* verwendet. Es wird benötigt, damit die PickMeUp-Entity funktioniert.
Das DCL-Plugin befindet sich auf der DVD in folgendem Verzeichnis:
[DVD-Laufwerk]:**Altamira\Editor-Dateien**

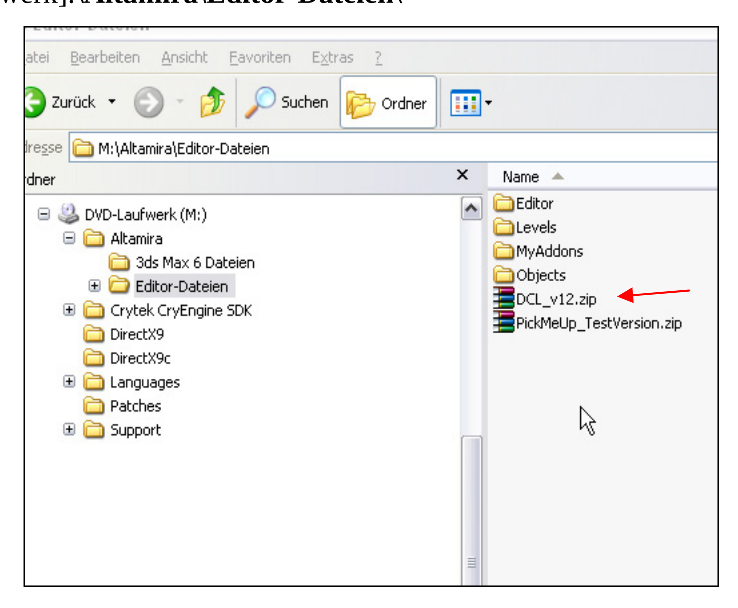


Abbildung 42: DCL-Plugin (siehe Pfeil)

Zur Installation muss die in der obigen Abbildung mit einem Pfeil markierte zip-Datei einfach in das Stammverzeichnis des Spiels **FarCry** entpackt werden.

Alle benötigten Dateien werden so an die richtigen Stellen kopiert. Bereits vorhandene Dateien können überschrieben werden.

Für nähere Informationen ist in der zip-Datei ein ReadMe enthalten, das nähere Hinweise zur Installation gibt.[35]

5.9 PickMeUp

Im gleichen Verzeichnis wie auch das DCL-Plugin liegt die Datei **PickMeUp_Testversion.zip.** Diese Dateien werden ebenfalls für die experimentelle *Beleuchtungssituation 4* benötigt.

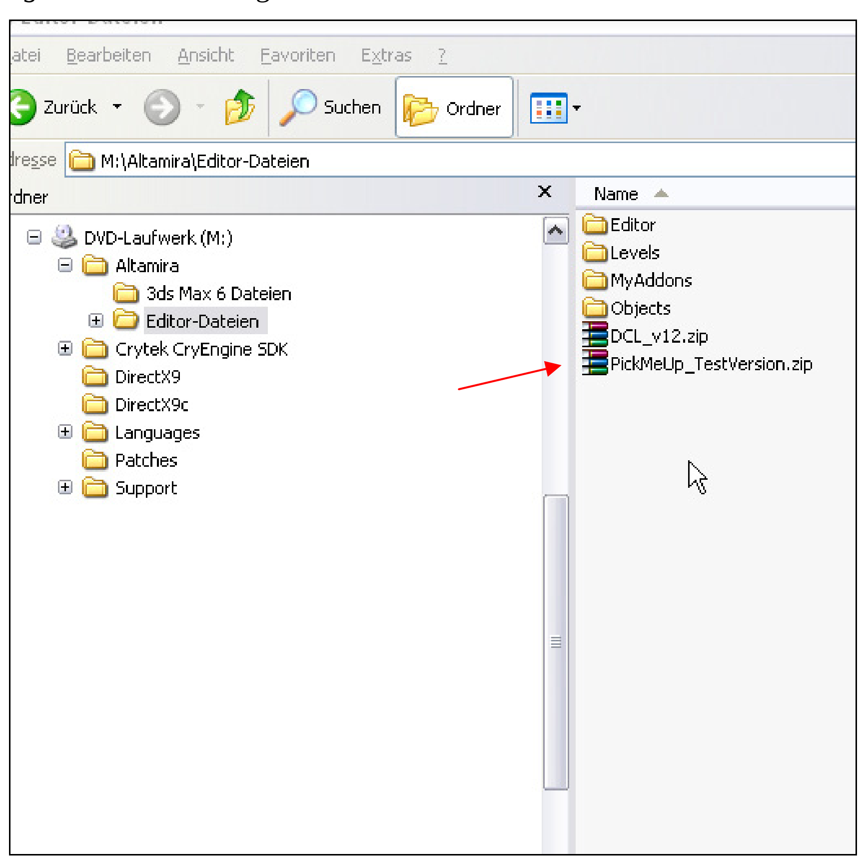


Abbildung 43: PickMeUp

Zur Installation muss die zip-Datei in das Stammverzeichnis von **FarCry** entpackt werden.

Bereits vorhandene Dateien können überschrieben werden.

Für weitere Informationen liegt der zip-Datei ein ReadMe bei.

---

[35] Für weitere Informationen über FarCry- DCL siehe auch: http://web.telia.com/~u31225218/fc/dcl/.

5.10 Der Sandbox-Editor

Der Editor wird mit dem Spiel FarCry installiert. In Abhängigkeit davon, in welches Verzeichnis das Spiel installiert wurde, ist der Editor unter folgendem Dateipfad zu finden:

*Laufwerk:\[Installationsverzeichnis-FarCry]**Bin32\.***

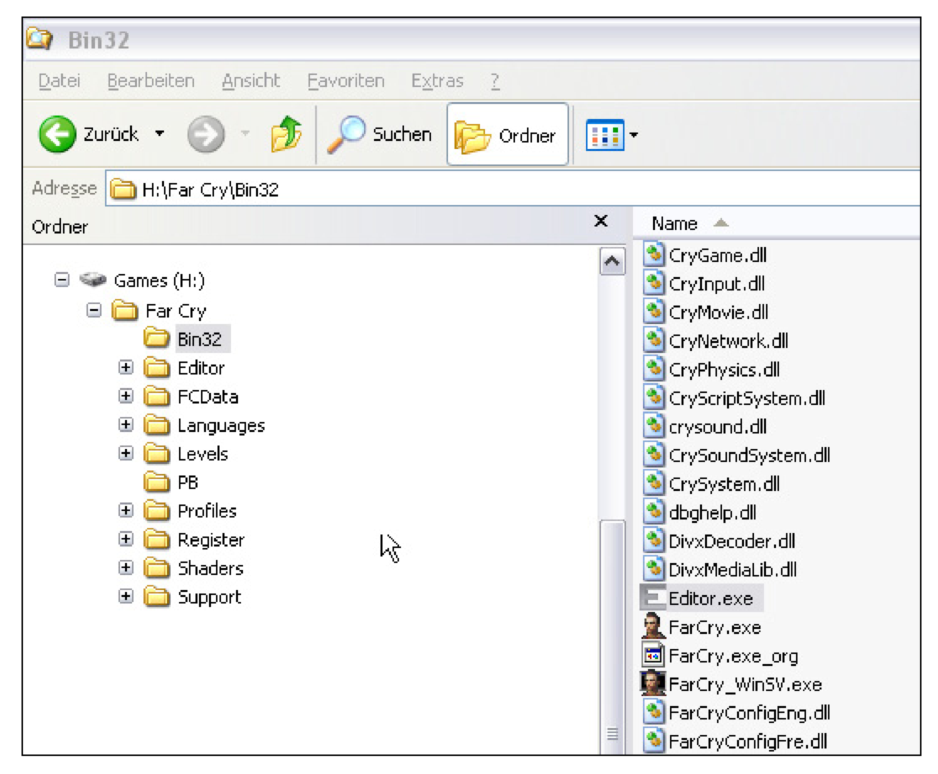


Abbildung 44: Der Sandbox-Editor im Installationsverzeichnis

Zur besseren Zugänglichkeit kann per Rechtsklick auf die Editor.exe eine Verknüpfung auf dem Desktop erstellt werden.

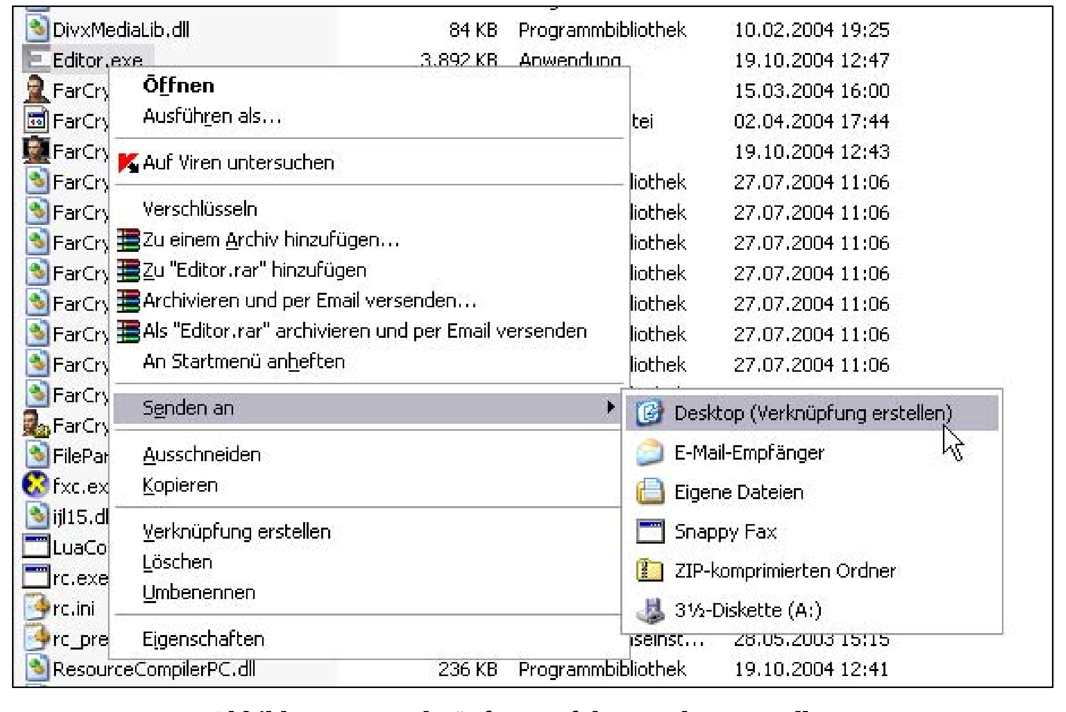


Abbildung 45: Verknüpfung auf dem Desktop erstellen

6 Konfiguration und Verwendung des Sandbox-Editors

Damit die beiliegenden Levels im Editor betrachtet werden können gilt es, den Editor richtig zu konfigurieren. Dies soll im Folgenden erläutert werden.

Der Editor übernimmt die Darstellungseinstellungen des Spiels. Um die beigelegten Beleuchtungssituationen in voller Qualität betrachten zu können, sollte 1) geeignete Hardware verwendet und 2) Vor dem Start des Editors das FarCry-Konfigurationsprogramm ausgeführt werden.

Dieses befindet sich unter (Default-Einstellung):

Start → Programme → Ubisoft → FarCry → **Far Cry konfigurieren**

Damit alle Details sowie die dynamische Beleuchtung richtig dargestellt werden sollten alle Werte wie folgt eingestellt werden:



Abbildung 46: FarCry konfigurieren: Spieleoptionen



Abbildung 47: FarCry konfigurieren: Grafikoptionen

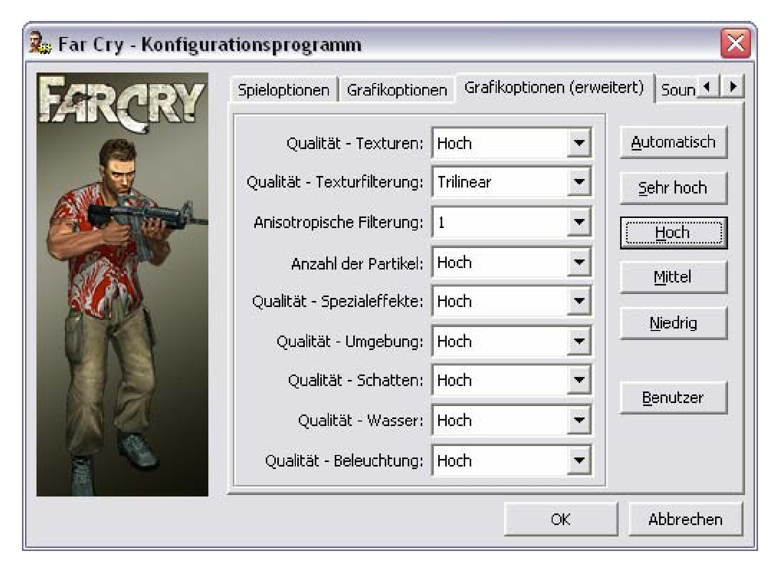


Abbildung 48: FarCry konfigurieren: Grafikoptionen (erweitert)

Die angegebenen Werte sind minimale Einstellungen um die beigelegten Ergebnisse in der beabsichtigten Qualität betrachten zu können. Selbstverständlich können auch höhere Qualitätseinstellungen vorgenommen werden. Allerdings gilt zu beachten, dass in der mittleren sowie der niedrigen Qualitätsstufe die dynamischen Lichtquellen nicht dargestellt werden!

Auf der FarCry-DVD befindet sich ein Ordner „Altamira“. Dieser enthält alle erforderlichen Dateien, sowohl für den Editor, als auch für 3ds Max.
Folgende Ordner müssen in das Stammverzeichnis von FarCry kopiert werden:
(wobei Stammverzeichnis heißt:
[Laufwerk oder Partition]:\[Name des Spielverzeichnisses (FarCry o.ä.)]\ )

1) [DVD-Laufwerk]:**Altamira\Editor-Dateien\Objects**
2) [DVD-Laufwerk]**:\Altamira\Editor-Dateien\Editor**
3) [DVD-Laufwerk]**:\Altamira\Editor-Dateien\Levels**

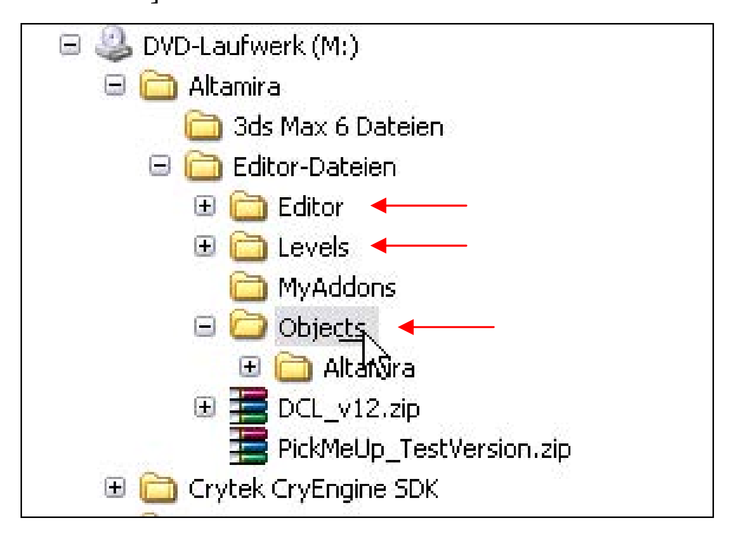


Abbildung 49: Die zu kopierenden Verzeichnisse

Bereits vorhandene Verzeichnisse können dabei überschrieben werden.

Das Stammverzeichnis (hier: H:\FarCry\) sollte nun in etwa so aussehen:

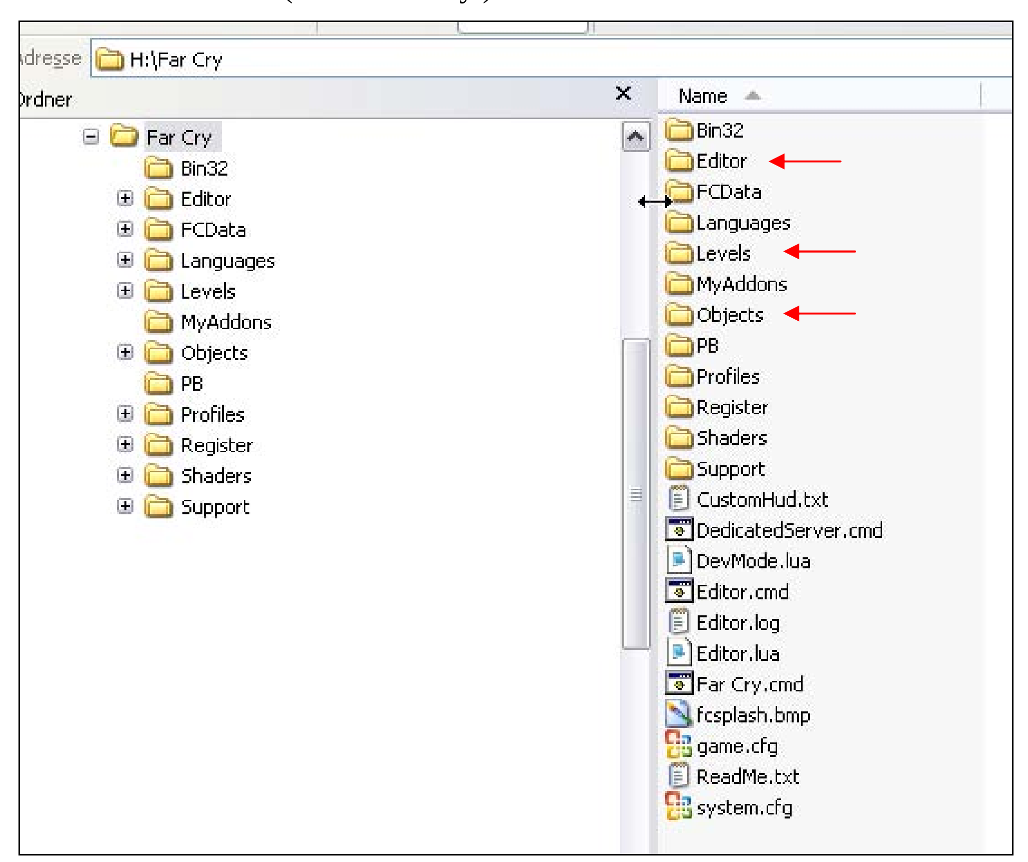


Abbildung 50: Stammverzeichnis des Spiels nach Kopieren (siehe Markierungen)

Der **Objects**-Ordner enthält das Modell der Höhle im proprietären cgf-Format des Editors und der **Levels**-Ordner enthält nun ein zusätzliches Verzeichnis namens **Altamira-Final**. In diesem liegen die vier Beleuchtungssituationen.
Damit diese vom Benutzer nachvollzogen und verändert werden können, werden im Folgenden einige Grundkenntnisse über den Umgang mit dem Editor vermittelt.

6.1 Grundlegende Bedienung

Nachdem der Editor nun fertig konfiguriert ist, werden nun einige grundlegende Kenntnisse zur Bedienung vermittelt.

Beim ersten Start des Editors erwartet den Benutzer folgender Anblick:

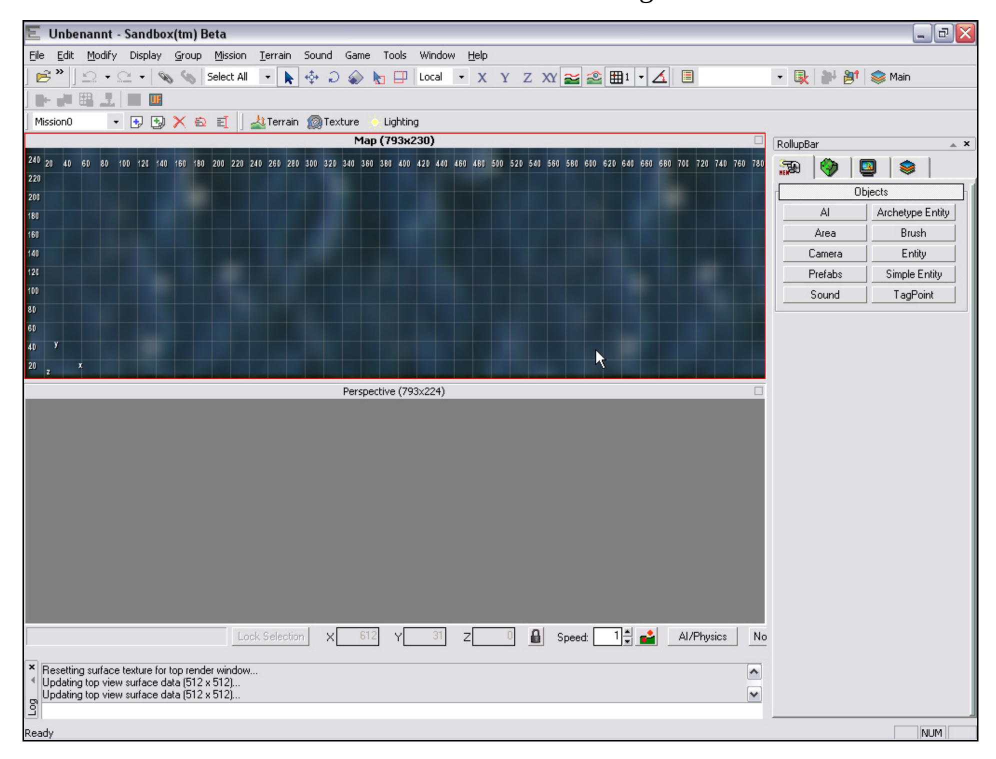


Abbildung 51: Sandbox-Editor beim ersten Start

Per Doppelklick auf die Kopfzeile eines Ansichtsfensters lässt sich die jeweilige Ansicht maximieren:

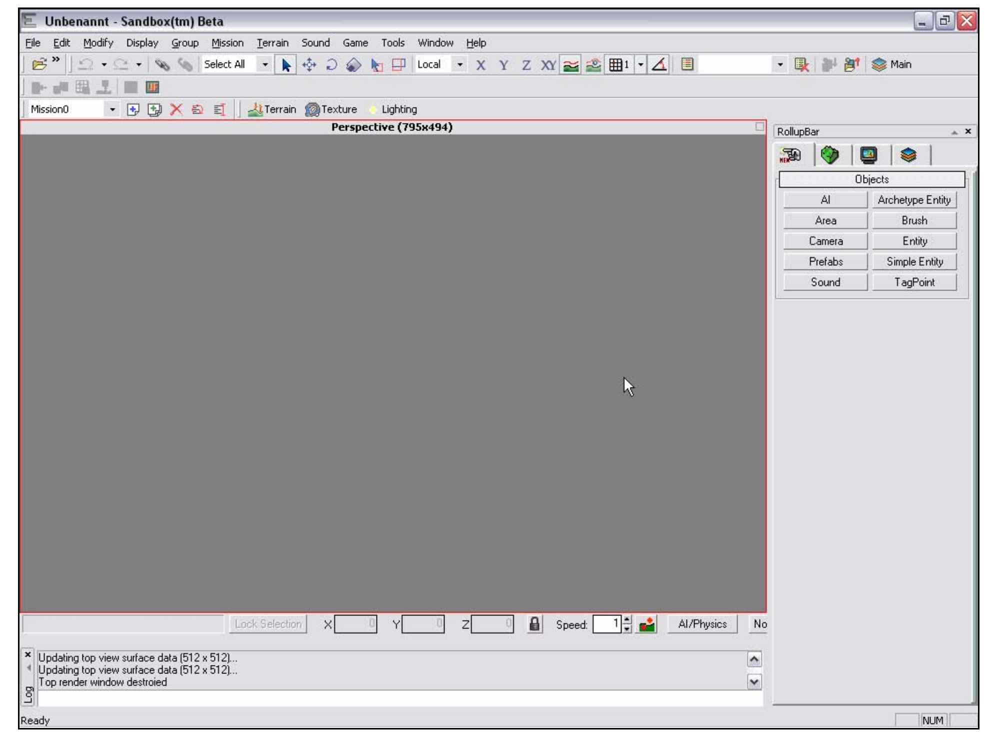


Abbildung 52: Maximierte Perspektiv-Ansicht

Mit einem Klick auf dieses Symbol kann ein Level geöffnet werden.

Die weiteren Schritte sollen an einem konkreten Beispiel erläutert werden. Dazu öffnen wir mit einem Klick auf obiges Symbol den *Datei öffnen*-Dialog:

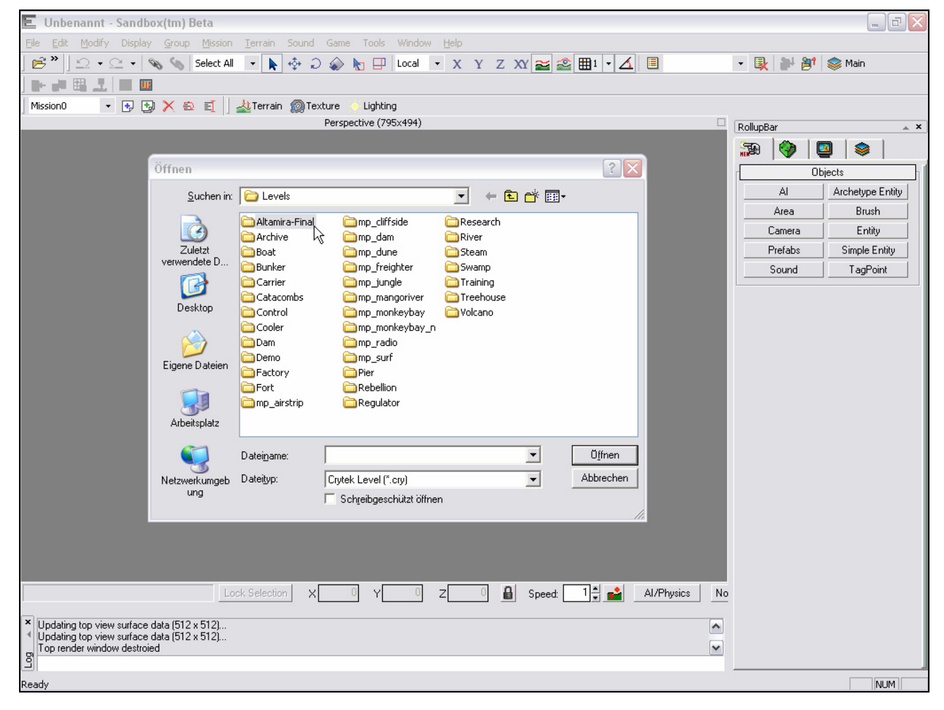


Abbildung 53: Öffnen eines Levels

Öffnen Sie folgende Datei: **Altamira-Final\Lagerfeuer\lagerfeuer.cry**.

Die erscheinende Fehlermeldung kann mit einem Klick auf **Close** geschlossen werden.

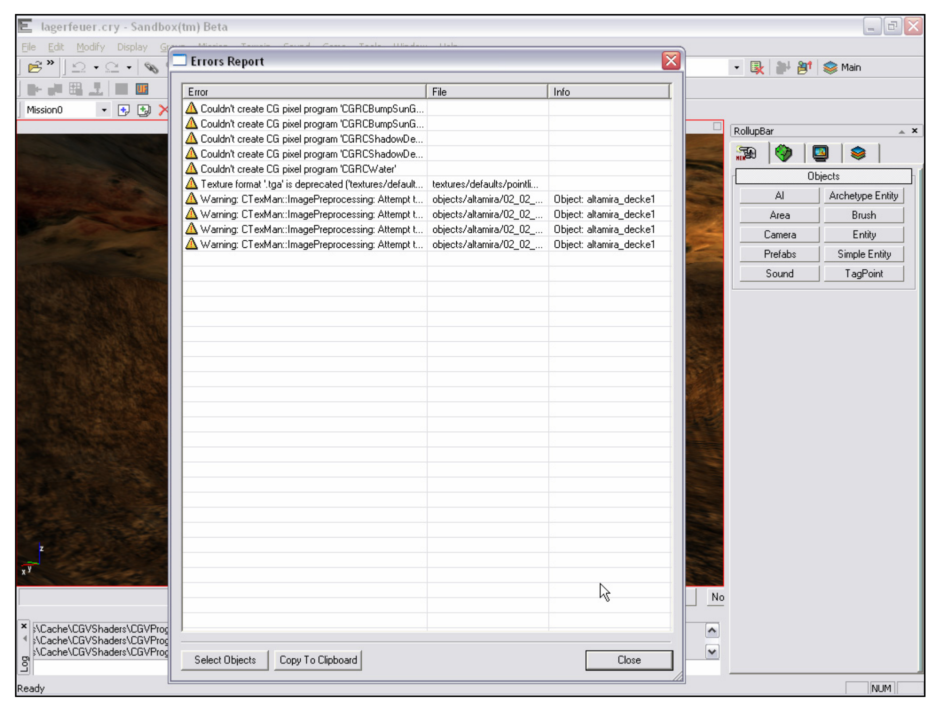


Abbildung 54: Fehlermeldung

Das geladene Level präsentiert sich folgendermaßen:

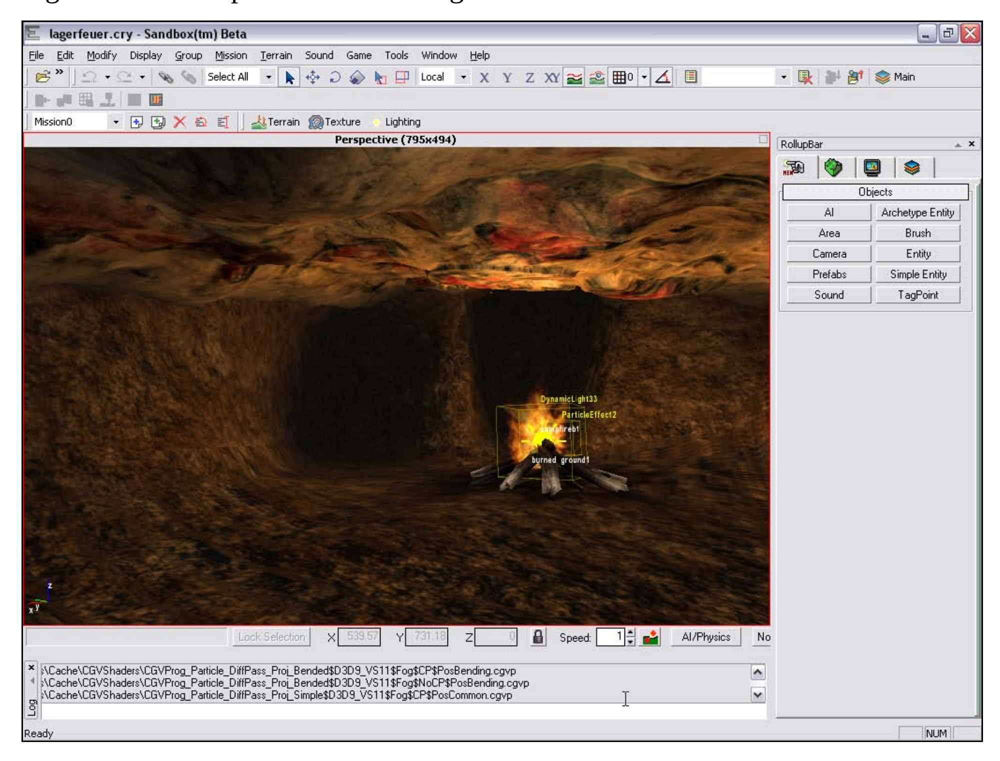


Abbildung 55: Levelansicht

6.2 Navigieren im Editor

Bei gedrückter **rechter** Maustaste kann man die Blickrichtung durch Bewegen der Maus verändern. Die Default-Einstellung ist: **W** – Vorwärts, **A** – Links, **D** – Rechts, **S** – Zurück. Mit der **linken** Maustaste kann man Objekte markieren. Mit dem **Mausrad** kann hinein- oder herausgezoomt werden.

Klicken Sie auf den Button *Entity*, rechts im Bild:

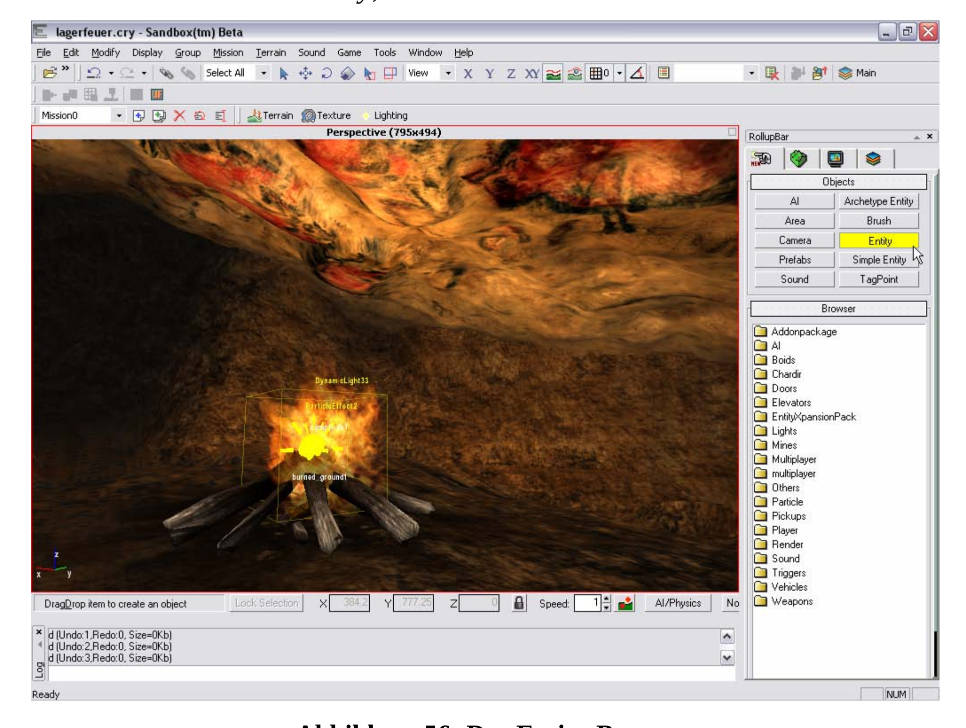
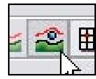


Abbildung 56: Der Entity-Button

Klicken Sie auf dieses Symbol in der oberen Leiste: 

Klicken Sie auf den Ordner *Lights* um ihn aufzuklappen und klicken Sie dann auf den Eintrag *DynamicLight*. Bei gehaltener linker Maustaste ziehen Sie das *DynamicLight* in das Ansichtsfenster.

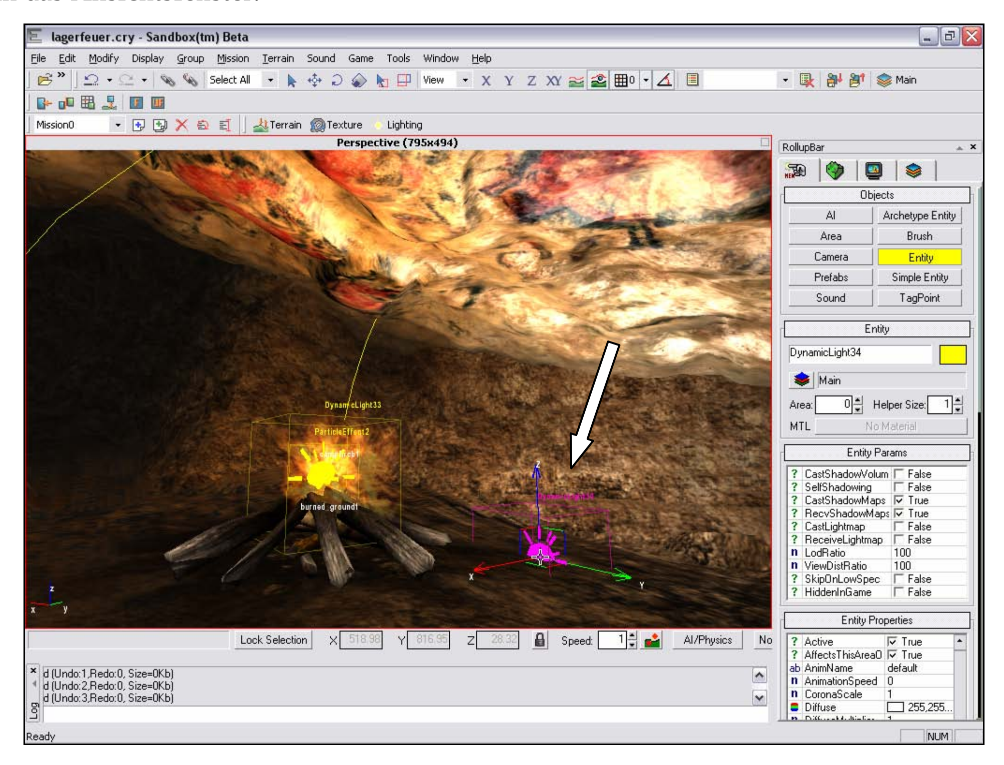
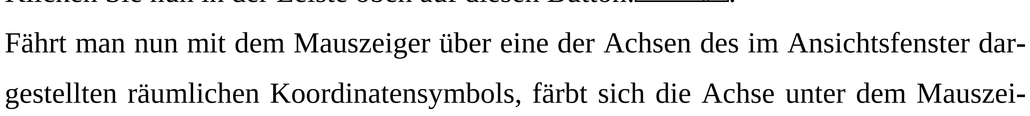


Abbildung 57: Eingefügtes *DynamicLight*

Am Mauszeiger „hängt“ nun das eingefügte Licht. Mit einem Klick auf die linke Maustaste kann die Lichtquelle vorerst an einer beliebigen Stelle platziert werden.

Klicken Sie nun in der Leiste oben auf diesen Button: .

Fährt man nun mit dem Mauszeiger über eine der Achsen des im Ansichtsfenster dargestellten räumlichen Koordinatensymbols, färbt sich die Achse unter dem Mauszeiger gelb.

In diese Richtung kann man das Objekt nun verschieben. Klicken Sie z.B. auf die blaue Z-Achse, halten Sie die Maustaste gedrückt und bewegen Sie die eingefügte Lichtquelle nach oben oder unten.

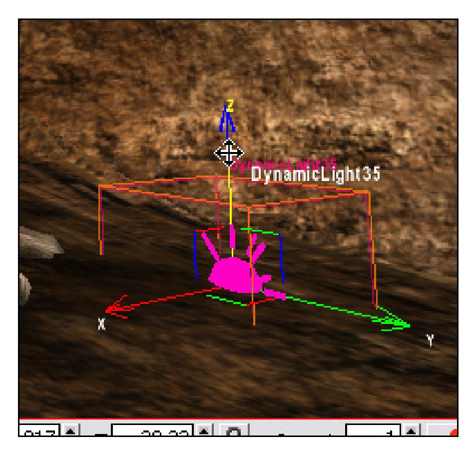


Abbildung 58: Objekte verschieben

Klickt man auf das Viereck zwischen zwei Achsen, kann man das Objekt frei bewegen.
Um die Lichtquelle jetzt in Ihrer Intensität und Reichweite zu verändern geht man so vor:
Markieren Sie die eingefügte Lichtquelle.
Rechts im Bild, im Rollout *Entity Params*, setzt man ein Häkchen bei *CastShadowVolume* und entfernt die Häkchen bei *CastShadowMap* und *ReceiveShadowMap*.[36]

Das Ergebnis sollte so aussehen:

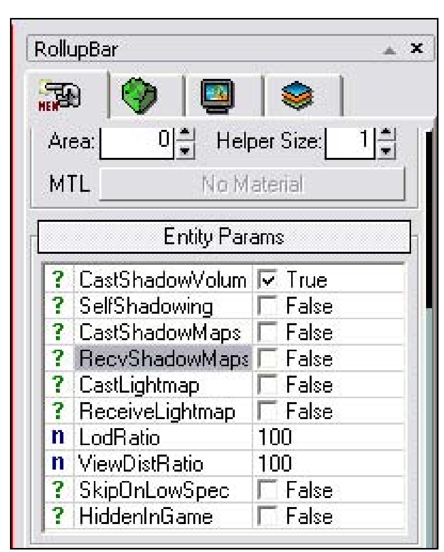


Abbildung 59: Entity Params

[36] ShadowMaps werden hier nicht verwendet, da wir dynamische Beleuchtung brauchen, ShadowMaps aber, ähnlich wie LightMaps, vorberechnet werden.

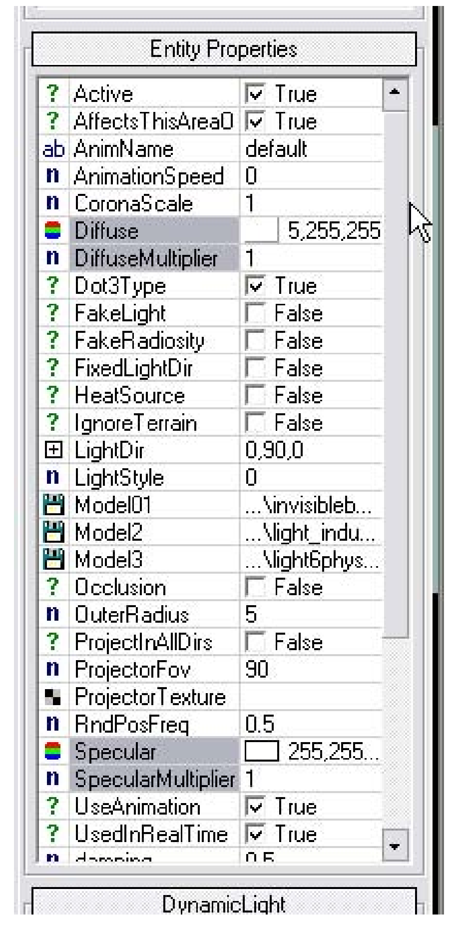


Abbildung 60: Entity Properties

Weiter unten, in den *Entity Properties*, kann, unter anderem, über die Punkte *Diffuse* und *DiffuseMultiplier*, die Lichtfarbe und der Grad der Wirkung des Lichts auf die umliegenden Texturen verändert werden. Analog gilt dies für *Specular* und *Specular-Multiplier*, nur, dass sich Veränderungen hier nur auf die hellen Glanzlichter der Lichtquelle auswirken.

Je höher der Wert des jeweiligen *Multipliers* ist, desto größer die Auswirkung der *Diffuse*-Farbe auf die Texturen. Der Default-Wert für die *Diffuse*-Farbe ist weiss.

Unter *LightStyle* kann die Art der Beleuchtung eingestellt werden (flackerndes Licht usw.).

Ein weiterer wichtiger Parameter ist *OuterRadius*, hier kann die Reichweite der Lichtquelle eingestellt werden. Auch hier gilt, je höher der Wert, desto größer die Reichweite.

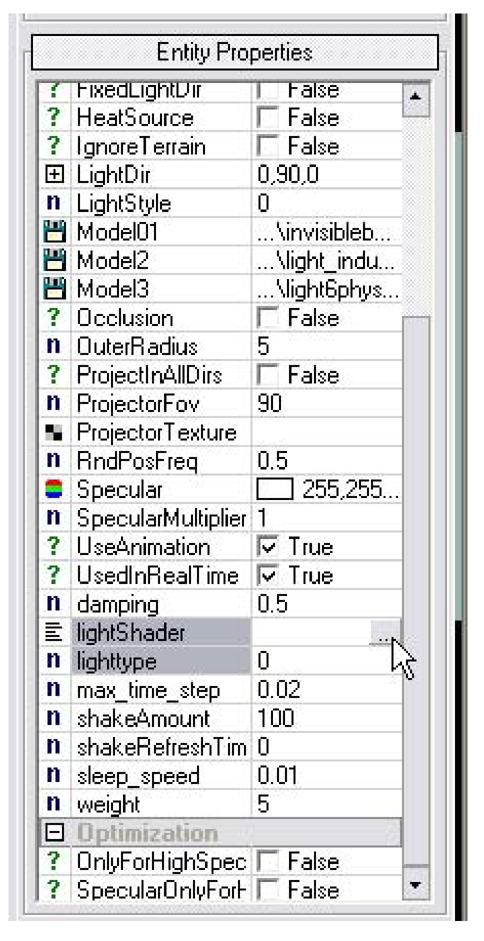


Abbildung 61: Fortsetzung Entity Properties

Die letzten wichtigen Punkte sind *lightShader* und *lighttype*. Unter *lightShader* können verschiedene, vorgefertigte *Shader* ausgewählt werden. Hier findet man sich bewegende Lichtquellen, flackernde Lichtquellen und vieles mehr. Die Bezeichnungen der *LightShader*-Programme beginnen in den meisten Fällen mit *Light-*. Zusätzlich kann man vorhandene *Shader* anpassen oder komplett neue, eigene kreieren. Der Punkt *lighttype* beeinflusst das Erscheinungsbild der Lichtquelle. Default ist „0“. Der Wert „1“ scheint nicht belegt zu sein, „2“ ist eine an zwei Drähten hängende Neonröhre und „3“ die bereits vorgestellt hängende runde Deckenleuchte. Wichtig ist hierbei, nach jeder Änderung der Parameter den *Reload Script*-Button zu betätigen, damit die Änderungen korrekt übernommen werden.

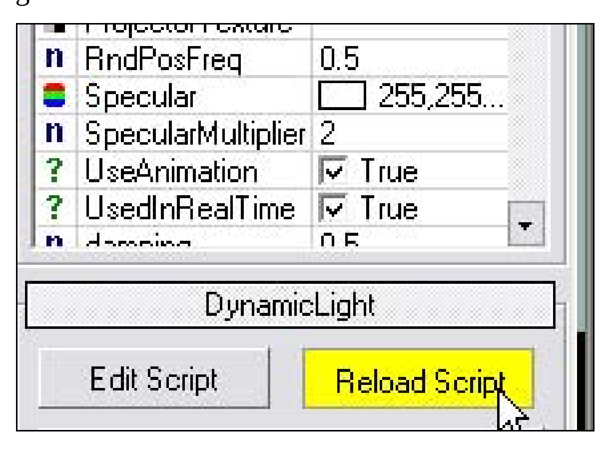


Abbildung 62: Der Reload Script-Button

Soweit die wichtigsten Parameter und Handgriffe, um zusätzliche Lichtquellen zu platzieren, zu bewegen und in ihrer Erscheinung zu verändern. Für weitere Informationen zur Bedienung des Editors liegt dieser Arbeit das offizielle *Crytek CryEngine Sandbox Manual v1.1* im pdf-Format bei.

7 Quellen- und Literaturverzeichnis

Literatur

Prähistorischer Malerei:

Aellen, Villy: *Die Höhlen Europas: Über 400 erschlossene Höhlen, Erklärung, Lage, Besonderheiten* / Villy Aellen; Pierre Strinati. – 1. Aufl. – München, Bern, Wien: BLV Verlagsgesellschaft, 1977.

Beltrán¸ Antonio: *Altamira* ... Mit Aufnahmen von Pedro A. Saura Ramos. Hrsg. und mit
einem Vorw. von Gerhard Bosinski. Sigmaringen: Thorbecke,1998.

Biedermann, Hans: *Höhlenkunst der Eiszeit – Wege zur Sinndeutung der ältesten Kunst Europas*. Köln: DuMont, 1984.

Lorblanchet, Michel: *Höhlenmalerei: Ein Handbuch*. Speläothek 1, Sigmaringen, 1997.

Pastoors, Andreas und Weniger, Gerd-Christian: *Bilder im Dunkeln: Höhlenkunst der Eiszeit - Die Sammlung Wendel*. Hrsg. v. Andreas Pastoors und Gerd-Christian Weniger, mit einem Vorwort von Paul G. Bahn. Mettmann: Neanderthal Museum, 2004.

Pastoors, Andreas und Gerd-C.Weniger: *Höhlenkunst und Raum – Archäologische und architektonische Perspektiven* / Neanderthal Museum. Hrsg. Von Andreas Pastoors und Gerd-Christian Weniger. Mettmann: Neanderthal Museum, 2003.

Ucko, Peter J. und Andrée Rosenfeld: *Felsbildkunst im Paläolithikum*. München: Kindler,1967.

3ds Max und CryEngine

Crytek, *CryEngine® Sandbox, Far Cry™ Edition, User Manual v1.1*. Coburg: Crytek GmbH, 2003. (Liegt der Arbeit als pdf-Datei bei).

Discreet, *3ds maxTM Version Vier – Lehrgänge*. 2001 Autodesk, Inc.

Immler, Christian: *3ds max 5 : Produktivität, Realismus, Dynamik* / [Christian Immler]. Burkhard Müller. - München/Germany : Markt und Technik, 2003.

Winkler, Peter: *Jetzt lerne ich 3D-Design mit 3D Studio Max 3*. München: Markt&Technik Verlag, 2000.

Virtuelle Realität

Kate Fernie and Julian D. Richards, *Creating and using Virtual Reality – A Guide for the Arts and Humanities*. Arts and Humanities Data Service. Oxford: Oxbow Books, 2003.

Internetquellen

3D-Programme:

Webseite:
Alias-Wavefront – Maya Produktseite. 04.02.2005.
http://www.alias.com/eng/products-services/industries/feature_film/index.shtml

Webseite:
Discreet – 3ds Max. 10.02.2005. http://www4.discreet.com/3dsmax/.

Webseite:
Maxon – Cinema4D in Film und TV. 04.02.02005.
http://www.maxon.de/pages/dyn_files/dyn_htx/htx/sol_movie_d.html.

Modding-Foren

Webseite:
Crymod. The Crytek Modding Portal. 09.02.2005. http://www.crymod.com.

Onlineposting:
Aarbro. “Carry entity, test version released“. Online posting. 01.12.04 The Crytek Modding Portal. 09.02.05.
http://www.crymod.com/index.php?templateid=showposts&fid=10&tid=1497 (Liegt der Arbeit als pdf-Datei bei).

Webseite:
FarCry-DCL. 11.02.2005. http://web.telia.com/~u31225218/fc/dcl/.

Webseite:
FarCry-Editor, 09.02.2005. http://www.farcry-editor.de.

3D-Engines

Webseite:
Crytek – CryEngine Featurelist. 04.02.2005.
http://www.crytek.de/technology/index.php?sx=cryengine (Liegt der Arbeit als pdf-Datei bei).

Webseite:
DevMaster.net. Unreal Engine 2 – Features. 04.02.05.
http://www.devmaster.net/engines/engine_details.php?id=39 (Liegt der Arbeit als pdf-Datei bei).

Webseite:
Genesis3D – 3D Engine. 04.02.2005. http://www.genesis3d.com/.

Webseite:
Irrlicht-Engine – 3D-Engine. 04.02.2005. http://irrlicht.sourceforge.net/index.html.

Webseite:
OGRE-Engine – 3D-Engine. 04.02.2005. http://www.ogre3d.org/.

Museen

Webseite:
Museum von Altamira – Abzeichnung der polychromen Decke. 04.02.2005.
http://museodealtamira.mcu.es/ingles/museo/images/pinturas.gif

Webseite:
Museum von Altamira – Homepage. 04.02.2005. http://museodealtamira.mcu.es.

Sonstige

Webseite:
Nvidia – Performance GeForceFX. 10.02.05:
http://www.nvidia.com/page/fx_5700.html (Liegt der Arbeit als pdf-Datei bei).

Webseite:
Nvidia – Performance Geforce 6800Ultra. 14.02.05
http://www.nvidia.de/page/geforce_6800.html (Liegt der Arbeit als pdf-Datei bei).

Webseite:
Nvidia – Texture Tools. 08.02.2005.
http://developer.nvidia.com/object/nv_texture_tools.html.

Webseite:
Ubisoft – FarCry – CryEngine SDK. 11.02.2005.
http://www.ubi.com/DE/Downloads/Info.aspx?dlId=594.

8 Erklärung

Ich versichere, dass ich die von mir vorgelegte Magisterarbeit selbstständig und ohne unzulässige Hilfe angefertigt, die benutzten Quellen, einschließlich der Quellen aus dem World Wide Web, und die Hilfsmittel vollständig angegeben und die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem World Wide Web im Wortlaut oder dem Sinn nach entnommen sind, in jedem Einzelfall als Entlehnung kenntlich gemacht habe.

______________________	________________________
Datum	Unterschrift